

LINGUAGEM BASIC MSX

gradiente

LINGUAGEM
BASIC
MSX

gradiente

901-01930-000

06-2647-001

COLEÇÃO MSX

LINGUAGEM BASIC MSX

4ª EDIÇÃO
1986

gradiente

Editora Aleph

© 1986 GRADIENTE ELETRÔNICA S.A.

Este livro foi elaborado pela equipe técnica de redação e arte da Editora Aleph com exclusividade para a Gradiente Eletrônica S.A. a partir de originais técnicos fornecidos pela Srta. Denise Santoro Cruz em agosto de 1985. 4ª Edição revista e ampliada.

COORDENAÇÃO EDITORIAL: Pierluigi Piazzi

EDITORAÇÃO E REDAÇÃO FINAL: Nancy Mitie Ariga e Renato da Silva Oliveira

COPY-DESK: Regina Brito de Assumpção

PRODUÇÃO EDITORIAL: Betty F. Piazzi e Rosa K. Fromer

ILUSTRAÇÃO E ARTE: Ana Lúcia Antico, Marcelo da Rocha Ciambra, Mário Dimov Mastrotti,

CAPA: Cassiano Roda e Fátima Rossini





ALEPH Publicações e Ass. Ped. Ltda.
Av. Brig. Faria Lima, 1451 - Conj. 31
01451 - São Paulo - SP
Tel.: (011) 813-4555 - 212-4917

GRADIENTE ELETRÔNICA S.A.
R. Henrique Monteiro, 40
05423 - São Paulo - SP
Cx. Postal 30318
Tel.: (011) 814-2299

CIP-Brasil. Catalogação-na-Publicação
Câmara Brasileira do Livro, SP

Cruz, Denise Santoro.
C961L Linguagem BASIC MSX / Denise Santoro Cruz. — 4ª ed. — São Paulo: Aleph, 1986.

(Coleção MSX)

1. BASIC (Linguagem de programação para computadores) 2. Microcomputadores — Programação I. Título.

17. CDD-651.8
18. -001.642
85-1518 18. -001.6424

Índices para catálogo sistemático:

---

1. BASIC MSX: Linguagem de programação: Computadores: Processamento de dados 651.8 (17.) 011.6424 (18.)
2. Microcomputadores: Programação: Processamento de dados 651.8 (17.) 001.642 (18.)
3. MSX BASIC: Linguagem de programação: Computadores: Processamento de dados 651.8 (17.) 001.6424 (18.)


MSX

SUMÁRIO

INTRODUÇÃO

Este livro foi escrito tendo-se em vista a necessidade de consulta a uma fonte de referência para a linguagem BASIC MSX.

Ele é muito mais um dicionário (ou talvez uma "enciclopédia") do BASIC MSX, do que um curso de programação.

Para os iniciantes no mundo da computação, recomendamos a leitura prévia do DOMINANDO O EXPERT, da mesma editora, para uma familiarização inicial.

Para os que já conhecem algum "dialeto" BASIC, este livro representará uma valiosa fonte de consulta.

As dificuldades de adaptação são minimizadas pelo fato do BASIC MSX ser o mais próximo possível do que poderíamos chamar de BASIC PADRÃO.

A concepção da máquina, por sinal, reflete toda uma filosofia básica rumo à compatibilização.

Quando a iluminação a gás começou a ser substituída pela elétrica, cada fabricante tinha um padrão diferente de rosca para suas lâmpadas. Isto fazia com que uma residência com um determinado tipo de instalação só pudesse ser equipada com lâmpadas de uma única marca.

O mundo da informática sofreu da mesma "doença juvenil". Cada fabricante se sentiu no direito de conceber sua própria versão de uma dada linguagem ou sistema operacional.

O padrão MSX, já adotado por quase duas dúzias de fabricantes no mundo inteiro, veio para acabar com a "Babel" que se instaurou no mundo da computação.

Desta forma, um "software" desenvolvido para uma certa marca, serve com certeza em outras, desde que esta obedeça os padrões MSX.

Trata-se, portanto, de um primeiro passo em direção à maturidade de uma área tão nova e tão dinâmica, maturidade esta com a qual o usuário só tem a ganhar.

Obviamente surgem críticas a esta padronização, pois ela impediria uma eventual evolução. Na realidade, a escolha do melhor microprocessador de 8 bits (Z80) e 32K de ROM (o APPLE tem 12!) garantem um superdimensionamento do sistema mais que suficiente, para afastar para um futuro distante qualquer obsolescência.

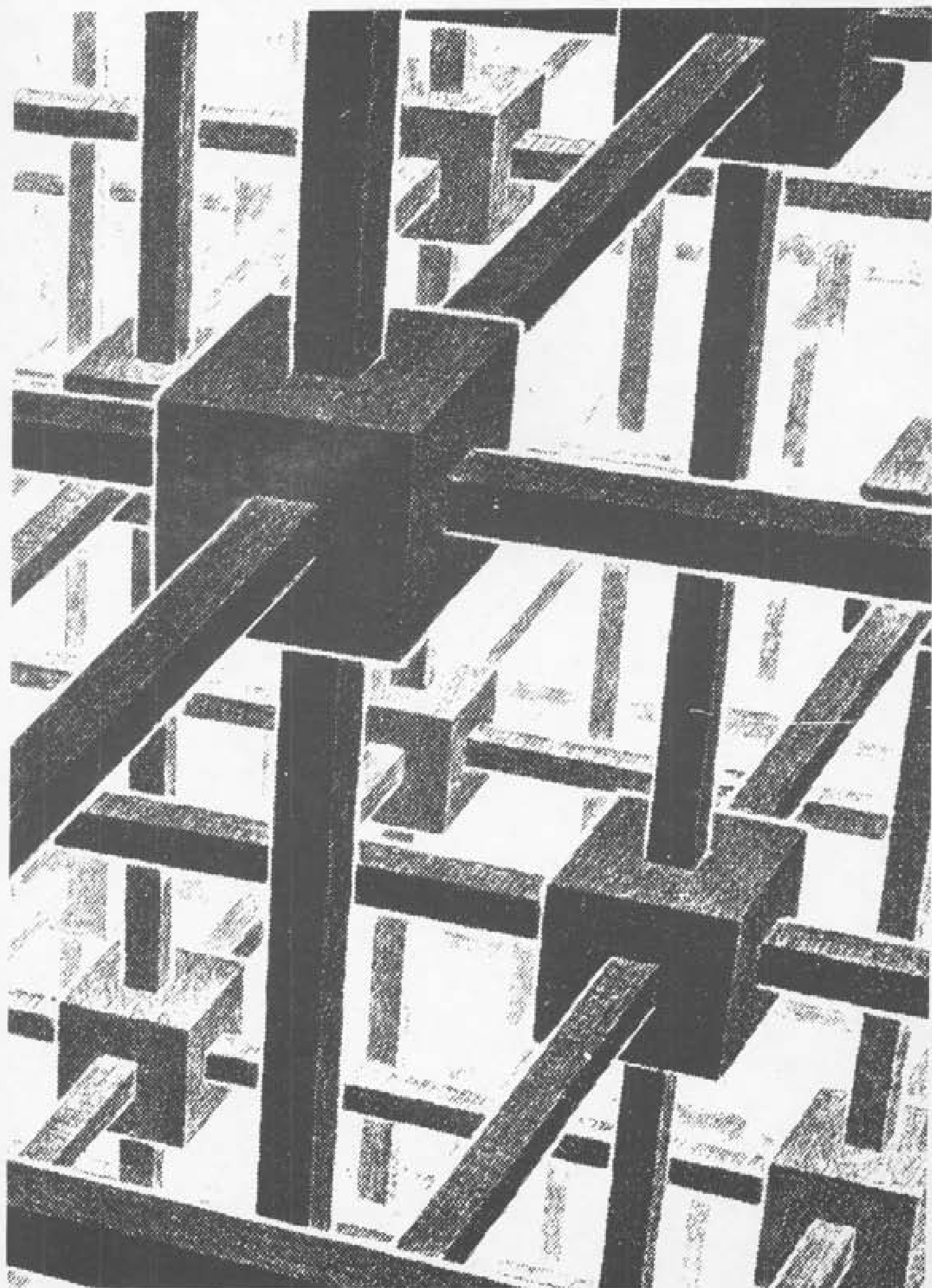
O fato da arquitetura de hardware do padrão MSX possuir 3 microprocessadores faz com que seu Basic, além de extremamente poderoso, seja também muito rápido. Em muitos computadores pessoais, um único microprocessador se encarrega de gerenciar os cálculos, o vídeo e o som. No padrão MSX o Z80 apenas envia informações aos processadores de vídeo e som que se encarregam de cumprir as tarefas indicadas, liberando-o para passar aos itens seguintes do programa.

A existência dos outros dois processadores, além da rapidez, proporciona vantagens adicionais: o comando DRAW, por exemplo, constitui-se numa verdadeira sub-linguagem gráfica, uma espécie de LOGO embutido no BASIC MSX. Da mesma forma, o comando PLAY permite comandar um poderoso sintetizador com 3 vozes. Podem-se efetuar alterações independentes na altura, intensidade, timbre e ataque de cada uma das vozes.

Com recursos deste quilate, torna-se simples a elaboração de programas em BASIC que de outra forma exigiriam árduas rotinas em Linguagem de Máquina.

Seja, portanto, bem-vindo ao maravilhoso mundo do BASIC-MSX.

PARTE I
A ESTRUTURA DO BASIC MSX



PARTE I

A ESTRUTURA DO BASIC MSX

Modos de Operação

Quando o computador é ligado, gera uma mensagem na tela:

```
MSX BASIC versão 1.1 Br
G r a d i e n t e
28815 bytes livres
Ok
```

Figura 1.1

Este "OK" significa que ele está apto a receber comandos. Eles podem ser introduzidos de duas formas:

MODO DIRETO: O comando é digitado sem ser precedido por um "número de linha". Ao teclarmos:

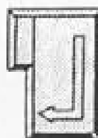


figura 1.2

ele é imediatamente executado. Seu resultado pode permanecer armazenado na memória, mas a ins-

trução em si é perdida.

MODO PROGRAMAÇÃO: Neste caso, cada linha contendo instruções é precedida por um número. O computador, inicialmente, armazena todas as linhas em sua memória (portanto, as instruções nelas contidas não se perdem após a execução) e, quando solicitado, as executa seqüencialmente.

As linhas do BASIC, no MSX, têm a seguinte estrutura:

Nº DA LINHA	INSTRUÇÃO	[: INSTRUÇÃO]	(CR)

onde:

Nº DA LINHA: deve ser um número inteiro entre 0 e 65529.

INSTRUÇÃO: deve ser uma instrução do BASIC MSX.

[: INSTRUÇÃO]: várias instruções podem ser digitadas numa mesma linha, desde que o total de caracteres nela contidos não ultrapasse 255. Os colchetes indicam complemento opcional.

(CR): *carriage return*, ou, simplesmente, *return*. É obtido pressionando-se a tecla da figura 1.2.

CONJUNTO DE CARACTERES

Os caracteres do MSX são os alfabéticos, os numéricos, os especiais, os gráficos e os do alfabeto europeu (Portugal também está na Europa).

* Caracteres alfabéticos: são as letras de A a Z, maiúsculas e minúsculas.
* Caracteres numéricos: são os algarismos de Ø a 9 (o zero é cortado para evitar confusões com a letra O).
* Caracteres especiais: além dos caracteres alfabéticos e numéricos, os seguintes caracteres especiais são reconhecidos pelo BASIC MSX:

CARACTERE	INTERPRETAÇÃO
	Branco (às vezes simbolizado por ␢)
=	Igual
+	Símbolo de adição
–	Símbolo de subtração
*	Símbolo de multiplicação
/	Símbolo de divisão
\	Símbolo de divisão inteira
^	Exponenciação
(	Abre parênteses
)	Fecha parênteses
%	Porcentagem
#	Número ou escopo

"	Aspas
!	Ponto de exclamação
;	Ponto e vírgula
$	Dolar ou cifrão
,	Vírgula
.	Ponto
'	Apóstrofo
:	Dois pontos
&	"E" comercial
?	Ponto de interrogação
<	Menor que
>	Maior que
_	Barra inferior
(BS)	*Back Space* (volta o cursor apagando)
(ESC)	*Escape*
(TAB)	Tabulador
(CR)	*Carriage Return*
(LF)	*Line Feed*

CONSTANTES

São valores usados durante uma execução e podem ser classificados em dois tipos: alfabéticos (*strings*) e numéricos.

CONSTANTE STRING

Uma constante *string* é uma seqüência de até 255 caracteres, digitada entre aspas.

CONSTANTE NUMÉRICA

As constantes numéricas são de 6 tipos:

* Inteiras: que não possuem ponto decimal. Seu valor pode estar entre -32768 e 32767.

* Ponto-fixo: são números reais positivos ou negativos, que podem conter o ponto decimal.

* Ponto-flutuante: são números positivos ou negativos representados na forma exponencial, semelhante à notação científica. Uma constante com ponto flutuante consiste em uma constante inteira ou de ponto-fixo (com ou sem sinal) seguida de uma letra E (ou D) e da quantidade de posições que o ponto decimal deve andar (se o deslocamento for para a esquerda, o número de posições é negativo). Por exemplo:

23.4E3 = 23400
419E-5 = .00419

As constantes de ponto flutuante podem assumir valores entre 10^{-64} e 10^{63}.

* Hexadecimal: números hexadecimais, precedidos por &H.

* Octal: números octais, precedidos por &O.

* Binários: números binários, precedidos por &B.

As constantes numéricas podem ser processadas com precisão simples (6 algarismos significativos) ou dupla (14 algarismos significativos). O computador reconhece uma constante com precisão simples quando, no ponto flutuante, é usada a letra E, ou quando ela é seguida pelo ponto de exclamação (!). Por exemplo:

-5.007E3
417.3!

Se, na notação de ponto flutuante, for usada a letra D ao invés do E, ou se a constante for seguida de (#), ela é de precisão dupla. Por exemplo:

2.71838183D-19
3.141592#

Se nada disso for indicado, o BASIC MSX interpretará a constante como sendo de precisão dupla.

VARIÁVEIS

Variáveis são nomes que podem ser atribuídos ou associados a resultados de cálculos feitos nos programas.

O nome correspondente a uma variável poderá ser de qualquer tamanho, sendo que só os dois primeiros caracteres são significativos. Este nome pode conter letras e números, porém o primeiro caractere deve ser sempre alfabético.

Uma variável não pode ter um nome igual ao de um comando BASIC: isto inclui todos os comandos, os termos, os nomes de funções e as operações.

TIPOS DE VARIÁVEIS

Assim como as constantes, as variáveis também podem ser alfanuméricas (cujo nome deve sempre terminar por $) e numéricas.

As variáveis numéricas são de 3 tipos e também podem ser distinguidas pelo último caractere do nome:

* Variáveis inteiras (%): são aquelas que só podem conter valores inteiros. Por exemplo, Z%.

* Variáveis de precisão simples (!): são aquelas que possuem valores com até 6 dígitos significativos. Exemplo: PREÇO!

* Variáveis de precisão dupla (#): são variáveis que possuem valores com até 14 algarismos significativos. Exemplo: PI#.

Quando o nome de variável não termina por nenhum destes caracteres especiais ($, %, !, #), o BASIC MSX a interpreta como sendo numérica e de precisão dupla.

MATRIZES OU VARIÁVEIS INDEXADAS

Uma matriz é um conjunto de valores, todos associados a uma variável de mesmo nome. Cada elemento da matriz é subscrito por índices inteiros (ou representados por expressões inteiras). A quantidade de índices utilizados determina o número de dimensões da matriz e não pode superar 255. Por exemplo:

A(20) – Elemento de uma matriz de uma dimensão

C(20,3,14) – Elemento de uma matriz de três dimensões

OCUPAÇÃO DE MEMÓRIA

Cada *string* ocupa um número de bytes igual ao número de caracteres que ela contém, mais 3. As variáveis numéricas e cada elemento de matriz ocupam um número de bytes conforme a lista a seguir:

Inteira...........................	2 bytes
Precisão Simples...................	4 bytes
Precisão Dupla....................	8 bytes

CONVERSÃO DE PRECISÃO

Se o valor de uma constante numérica for atribuído a uma variável *string* (ou vice-versa) será explicitada uma mensagem de erro "Type Mismatch".

Se o valor de uma constante numérica de um certo tipo for atribuído a uma variável numérica de tipo diferente, ele será armazenado na memória do computador na forma determinada pelo tipo de variável. Por exemplo, o programa:

```
10 Z%=257.8131#
20 PRINT Z%
```

dará como resultado:

```
257
```

Note que a parte decimal foi truncada sem arredondamento.

Durante cálculos, operandos de tipos diferentes serão todos convertidos para o tipo de maior precisão, e nesse tipo será fornecido o resultado.

Os operadores lógicos convertem seus operandos em inteiros. Se eles caírem fora da faixa permitida aos inteiros, ocorrerá erro (*overflow*).

OPERADORES

Os operadores agem sobre constantes e variáveis contidas numa expressão, de maneira a produzir um valor único como resultados. Eles podem ser divididos em quatro categorias:

ARITMÉTICOS
RELACIONAIS
LÓGICOS
FUNCIONAIS

OPERADORES ARITMÉTICOS

Os operadores aritméticos, relacionados na ordem de prioridade, são:

OPERADOR	OPERAÇÃO	EXPRESSÃO EXEMPLO
^	Exponenciação	X ^ Y
–	Mudança de Sinal	–X
*, /	Multiplicação e Divisão	X * Y X / Y
\	Divisão Inteira	X \ Y
MOD	Resto da Divisão	X MOD Y
+, –	Adição e Subtração	X + Y X – Y

Para mudar a prioridade, ou seja, a ordem de execução, devemos usar parênteses.

Observação: Note que \ e MOD transformam seus operandos em inteiros e fornecem resultados inteiros, sendo que \ fornece o quociente e MOD o resto da divisão. Note também que o sinal de adição (+) pode ser usado para concatenar *strings*. Por exemplo:

"GRADI" + "ENTE" = "GRADIENTE"

OPERADORES RELACIONAIS

Estes operadores permitem comparações de valores:

OPERADOR	OPERAÇÃO	EXEMPLO
=	Igual	X = Y
<>	Diferente	X<>Y
<	Menor que	X < Y
>	Maior que	X > Y
<=	Menor ou igual	X <= Y
>=	Maior ou igual	X >= Y

Observação: Os operandos relacionados permitem também a comparação de *strings*. Esta comparação é feita entre os códigos ASCII de seus caracteres.

OPERADORES LÓGICOS

Estes operadores permitem a manipulação de bits, testes em relações múltiplas e operações de álgebra booleana. Eles estão listados em ordem de prioridade.

NOT

X	NOT X
1	Ø
Ø	1

AND

X \ Y	1	Ø
1	1	Ø
Ø	Ø	Ø

X AND Y

OR

X \ Y	1	0	
1	1	1	X OR Y
0	1	0	

XOR

X \ Y	1	0	
1	0	1	X OR Y
0	1	0	

EQV

X \ Y	1	0	
1	1	0	X EQV Y
0	0	1	

IMP

X \ Y	1	0	
1	1	0	X IMP Y
0	1	1	

Por exemplo, o resultado da instrução:

```
PRINT 60 XOR 240
```

será:

```
204
```

pois:

```
          60 = &B 00111100
         240 = &B 11110000
  60 XOR 240 = &B 11001100 = 204
```

FUNÇÕES

O BASIC MSX tem funções residentes, ou seja, funções cuja técnica de processamento já está gravada no seu sistema operacional, tais como LOG (logaritmo), SIN (seno), CSRLIN (cursor line), etc.

Além destas funções, o BASIC MSX admite outras concebidas pelo usuário. Neste caso ele deverá defini-las pela instrução DEF FN.

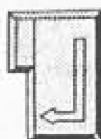
EDIÇÃO DE PROGRAMAS

O BASIC MSX dispõe de um editor cuja função é escrever, alterar e armazenar as linhas numeradas que constituem o programa.

Este editor permite mover o cursor por toda a tela, reescrever caracteres, apagá-los, inseri-los e armazenar linhas de BASIC na memória.

Uma linha de BASIC é aceita e armazenada na memória se precedida por um número inteiro entre 0 e 65529 e se contiver pelo menos um caractere diferente do espaço.

Para que ela seja mandada para a memória, após terminar sua digitação, devemos sempre teclar o RETURN.



Se digitarmos uma nova linha com o número de uma já armazenada, ao teclarmos RETURN, a segunda substitui a primeira, que é apagada. Isto ocorre também quando a nova linha tem só o número. Esta característica pode ser usada para apagar linhas.

Várias instruções podem ser colocadas numa mesma linha, desde que separadas por dois pontos (:). O total de caracteres contidos na linha, porém, não pode exceder 255.

Na tabela a seguir, são relacionadas as combinações de teclas que produzem efeitos especiais no movimento e funções do cursor, e seu equivalente (quando existir) nas teclas especiais.

Código		Combinações de Teclas	FUNÇÃO	Teclas Especiais
Dec.	Hex.			
02	02	CONTROL + B	Move o cursor até o começo da palavra anterior	
03	03	CONTROL + C	Dá um BREAK na espera do INPUT	
05	05	CONTROL + E	Apaga do cursor ao fim da linha BASIC	
06	06	CONTROL + F	Move o cursor até o começo da palavra seguinte	
07	07	CONTROL + G	Dá um "bip"	
08	08	CONTROL + H	Volta o cursor, apagando	BS
09	09	CONTROL + I	Move o cursor até a próxima tabulação	TAB
10	0A	CONTROL + J	Coloca o cursor na próxima linha (line feed)	
11	0B	CONTROL + K	Move o cursor até o canto superior esquerdo (HOME)	HOME
12	0C	CONTROL + L	Limpa a tela colocando o cursor em HOME	CLS
13	0D	CONTROL + M	Insere a linha BASIC na memória	
14	0E	CONTROL + N	Leva o cursor ao fim da linha	
18	12	CONTROL + R	Põe o cursor no modo inserção	INSERT
21	15	CONTROL + U	Apaga a linha interna	
24	18	CONTROL + X		SELECT
27	1B	CONTROL + [		ESC

Pressionando simultaneamente as teclas CONTROL + SHIFT + STOP, obtemos um RESET por software, ou seja, o Expert reinicializa as variáveis do sistema, deixando-as no estado em que se encontravam ao ligar o micro.

PARTE II
DICIONÁRIO DAS PALAVRAS RESERVADAS

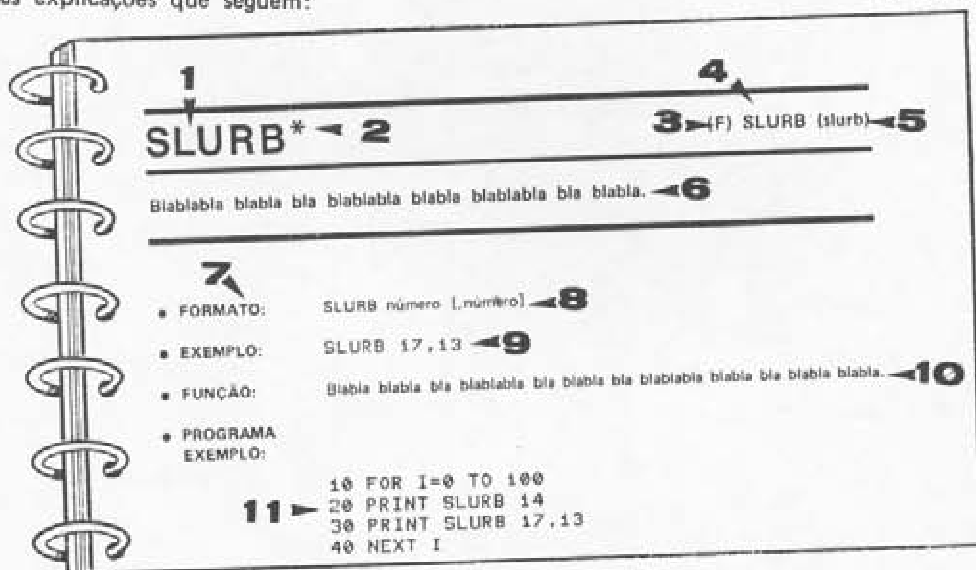


PARTE II

DICIONÁRIO DAS PALAVRAS RESERVADAS

CONVENÇÕES USADAS NO DICIONÁRIO

Antes de consultar o dicionário de palavras reservadas do BASIC MSX, leia atentamente as explicações que seguem:



① Nome do comando, instrução ou função. Trata-se de uma *palavra reservada* do BASIC MSX e não pode ser usada como nome de variável.

② Este asterisco indica que a instrução ou função assim marcada é usada por programadores mais experientes. Maiores detalhes sobre sua utilização podem ser encontrados no livro APROFUNDANDO-SE NO MSX.

③ Indica que a palavra reservada refere-se a uma função. Quando (F) for omitido, trata-se de uma instrução ou comando.

④ Palavra reservada.

⑤ Expressão completa, em inglês, da qual foi extraída a palavra reservada.

⑥ Explicação, resumida, da função que a palavra reservada tem no BASIC MSX.

⑦ Indica o formato (sintaxe) do comando ou função.

⑧ O que estiver entre colchetes [ ] é opcional, podendo ser eventualmente omitido.

⑨ Exemplo de formato, para mostrar uma forma sintaticamente correta.

⑩ Explicação mais detalhada da instrução ou função.

⑪ O programa exemplo deve ser, sempre que possível, digitado e rodado para melhor elucidação.

ABS

(F) ABS (absolute)

Fornece o valor absoluto de um número.

- **FORMATO:** ABS (argumento)
- **EXEMPLO:** B=ABS(-2)
- **FUNÇÃO:** Fornece o valor absoluto do argumento, ou da expressão X.
  ABS (X) = X para todo X maior ou igual a zero.
  ABS (X) = -X para todo X menor que zero.
- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA ABS
20 FOR F=1 TO 20
30 A=100-INT(RND(-TIME)*200)
40 PRINT A;".......";ABS(A)
50 NEXT F
```

ASC

Fornece o código ASCII do primeiro caractere de uma dada *string*.

- **FORMATO:** ASC (*string*)
- **EXEMPLO:** A=ASC("A")
- **FUNÇÃO:** Fornece o código ASCII (*American Standard Code for Interchange Information*) do primeiro caractere de uma *string* especificada. A *string* ou variável *string* deve ser colocada entre parênteses, não podendo ser nula. Caso contrário, ocasionará uma mensagem de erro.
- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA ASC
20 A$=INKEY$
30 IF A$="" THEN 20
40 PRINT"ASC(";A$;")=";ASC(A$),
50 GOTO 20
```

ATN

(F) ATN (arc tangent)

Fornece o valor do arco, em radianos, cuja tangente trigonométrica é igual ao argumento.

- FORMATO: ATN (argumento)

- EXEMPLO: A=ATN(1)

- FUNÇÃO: A função ATN fornece o arco-tangente (em radianos) do argumento, isto é, o ângulo cuja tangente é igual ao valor do argumento. O limite deste resultado está entre $-\pi/2$ e $\pi/2$. A expressão do argumento, pode ser qualquer número, mas o cálculo de ATN sempre será realizado em modo de precisão dupla.

- PROGRAMA EXEMPLO:

```
10 REM PROGRAMA ATN
20 CLS
30 SCREEN 2
40 FOR F=-128 TO 127
50 PSET(F+128,80-70*ATN((F)*6.2832/255)
)
60 NEXT F
```

AUTO

Gera, automaticamente, números de linhas a partir de um valor inicial especificado, com incremento também especificado.

- **FORMATO:** AUTO [número inicial da linha] [,incremento]
- **EXEMPLO:** AUTO 100,5
- **FUNÇÃO:** Este comando liga uma função automática de numeração de linha, para entrada de programas. Tudo o que você tem a fazer é introduzir as instruções reais do programa. Você pode especificar um número de linha inicial e um incremento para ser usado entre os números de linha ou então, simplesmente digitar AUTO e pressionar RETURN. Neste caso, a numeração de linha iniciará na 10 e usará incremento 10. Cada vez que você pressionar RETURN, o computador avançará para o número seguinte de linha.

Por exemplo:

COMANDO	NUMERAÇÃO DAS LINHAS DO PROGRAMA
AUTO	10, 20, 30 . . .
AUTO 5,5	5, 10, 15 . . .
AUTO 100	100, 110, 120 . . .
AUTO 100,25	100, 125, 150 . . .
AUTO 0	0, 10, 20 . . .
AUTO 10	10, 20, 30 . . .

Para desligar a função AUTO, pressione CTRL + C ou CTRL + STOP. Não será considerada a linha em que for teclado o CTRL + C. Quando o comando AUTO traz com o número da linha um asterisco (*), significa que a linha já está sendo usada. Se você não deseja reprogramar a linha, deverá pressionar CTRL + STOP para desligar a função.

- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA AUTO
20 PRINT"VOCE QUER ALTERAR O PROGRAMA A
 PARTIR  DA LINHA  100? (S/N)?"
30 A$=INKEY$
40 IF A$="S" THEN 90
50 IF A$="N" THEN 100
60 GOTO 30
90 AUTO 100,10
100 PRINT"VOCE NÃO ME ALTEROU."
```

BASE *

BASE (base)

Permite o acesso ao endereço inicial das tabelas utilizadas pelo processador de vídeo.

- **FORMATO:** BASE (número)
  BASE (número) = expressão

- **EXEMPLO:** BASE(10)

- **FUNÇÃO:** O comando BASE permite que o usuário leia e reescreva na memória o endereço inicial das tabelas utilizadas pelo VDP (Video Display Processor). Estas tabelas são utilizadas pelo processador de vídeo conforme o modo em que se está trabalhando. A expressão deve ser um valor inteiro entre 0 e 65535 e o número deve estar entre 0 e 19 (não se utilizando os valores 1, 3, 4 e 16). As tabelas, cujos endereços iniciais são especificados por BASE, estão relacionadas a seguir:

SCREEN	DESCRIÇÃO
0	0 – Tabela dos nomes utilizados no modo texto de 40 caracteres x 24 linhas 2 – Tabela do gerador de padrões utilizados no modo texto de 40 caracteres x 24 linhas
1	5 – Tabela dos nomes utilizados no modo texto de 32 caracteres x 24 linhas 6 – Tabela de cores utilizadas no modo texto de 32 caracteres x 24 linhas 7 – Tabela do gerador de padrões utilizados no modo texto de 32 caracteres x 24 linhas 8 – Tabela dos atributos de *sprites* utilizados no modo texto de 32 caracteres x 24 linhas 9 – Tabela dos padrões de *sprites* utilizados no modo texto de 32 caracteres x 24 linhas
2	10 – Tabela dos nomes utilizados no modo gráfico em alta-resolução 11 – Tabela de cores utilizadas no modo gráfico em alta-resolução 12 – Tabela do gerador de padrões utilizados no modo gráfico em alta-resolução 13 – Tabela dos atributos de *sprites* utilizados no modo gráfico em alta-resolução 14 – Tabela dos padrões de *sprites* utilizados no modo gráfico em alta-resolução
3	15 – Tabela dos nomes utilizados no modo policromático 17 – Tabela do gerador de padrões utilizados no modo policromático 18 – Tabela dos atributos de *sprites* utlizados no modo policromático 19 – Tabela dos padrões de *sprites* utilizados no modo policromático

- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA BASE
20 FOR F=1 TO 15:PRINT BASE(F):NEXT F
```

BEEP

Provoca um *beep* (sinal sonoro).

- **FORMATO:** BEEP
- **EXEMPLO:** BEEP
- **FUNÇÃO:** Utilizado para gerar um sinal sonoro. Produz o mesmo efeito que o CHR$ (7).
- **PROGRAMA EXEMPLO:**

```
0  REM programa BEEP
10 FOR I = 0 TO 19
20 BEEP
30 NEXT I
```

BIN$

(F) BIN$ (binary dollar)

Transforma um dado numérico em uma expressão binária na forma de uma *string*.

- **FORMATO:** BIN$ (X)

- **EXEMPLO:**

```
LPRINT BIN$(A)
```

- **FUNÇÃO:** O comando BIN$ transforma um dado numérico (constante, variável numérica ou variável de matriz numérica) em uma expressão binária na forma de uma *string*. Caso o número seja negativo, primeiro calcula-se o valor (em decimal) subtraindo-se de 65536, e a partir do resultado desta operação, faz-se a conversão para binário e a transforma posteriormente numa *string*.

```
PRINT BIN$(115)
1110011
PRINT BIN$(-19)
1111111111101101   ( 65536-19=65517 )
```

- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA BIN$
20 FOR F=0 TO 20
30 PRINT F;"......";BIN$(F)
40 NEXT F
```

BLOAD (binary load)

*BLOAD

Carrega um programa em linguagem de máquina e o executa (caso esta opção esteja ativada).

- **FORMATO:** BLOAD "nome do dispositivo [nome do arquivo]" [,R] [,deslocamento]
- **EXEMPLO:** BLOAD"CAS:EXEMPL",R
- **FUNÇÃO:** Carrega um programa em linguagem de máquina do dispositivo especificado, em geral "CAS:" (cassete). O nome do arquivo deve ter no máximo 6 caracteres (os excedentes serão desprezados). Se o nome do arquivo for omitido, o primeiro programa, em linguagem de máquina, que for encontrado, será carregado. Se R for especificado no comando, depois de carregado o programa, a execução começará automaticamente no endereço que havia sido indicado em BSAVE por ocasião da gravação. O programa será armazenado no trecho de memória especificado também em BSAVE pelos endereços inicial e final. Se o deslocamento está especificado, todos estes endereços são deslocados. O deslocamento deve ser um número inteiro entre -32768 e 65535.

- **PROGRAMA EXEMPLO:**

```
10 BLOAD "CAS:ALEPH"
20 CLS:FOR I=0 TO 41
30 PRINT HEX$(PEEK(60000!+I));" ";
40 FOR T=1 TO 90:NEXT T
50 NEXT I
60 FOR T=1 TO 500:NEXT T
70 DEFUSR=60006!
80 X=USR(0):GOTO 80
90 REM
100 REM USAR EM CONJUNTO COM O
        PROGRAMA BSAVE
```

BSAVE *

BSAVE (binary save)

Grava programas em código de máquina extraídos de uma área específica da memória.

- **FORMATO:** BSAVE "nome do dispositivo [nome do arquivo]", endereço inicial, endereço final, [endereço do início de execução]

- **EXEMPLO:** BSAVE"CAS:DEMO",&HA100,&HA2EF

- **FUNÇÃO:** Este comando é utilizado para gravar o conteúdo de uma área da memória em um dispositivo (em geral, "CAS:", onde CAS = cassete). O nome do arquivo deve ter no máximo 6 caracteres (os excedentes serão ignorados). O nome do arquivo pode ser também uma *string* nula. Os endereços, inicial e final, indicam a área da memória que deve ser gravada. Devem ser números inteiros entre -32768 e 65535. O endereço do início de execução indica o endereço no qual o programa deverá iniciar a execução quando for carregado de volta ao computador pelo comando BLOAD com a opção R ativada. Também deve ser um número inteiro entre -32768 e 65535.

- **PROGRAMA EXEMPLO:**

```
10 DATA 41,4C,45,50,48,FF,11,60,EA,1A
20 DATA FE,FF,C8,21,00,00,CD,4D,00,23
30 DATA E5,D5,11,BF,03,ED,52,D1,E1,38
40 DATA F1,21,FF,FF,2B,7C,B5,28,FB,13
50 DATA 18,DF
60 FOR I=0 TO 41
70 READ A$
80 POKE 60000!+I,VAL("&H"+A$)
90 NEXT I
100 DEFUSR=60006!
110 SCREEN 0
120 X=USR(0)
130 FOR T=1 TO 1000:NEXT T:CLS
140 BSAVE "CAS:ALEPH",60000!,60042!,6000
6!
```

CALL (call)

*CALL

Executa um comando contido num cartucho ROM.

- **FORMATO:** CALL nome do comando [, (argumentos)]

- **EXEMPLO:** CALL START

- **FUNÇÃO:** Serve para chamar um comando de uma memória auxiliar em cartucho. Do mesmo modo que o comando PRINT pode ser substituído por [?] e o comando REM pode ser substituído pelo ['] , o comando CALL pode ser substituído pelo sinal [_] .

- PROGRAMA EXEMPLO:

```
10 ' CALL
20 PRINT"Voce quer formatar seu diskette (s/n)?"
30 Z$=INPUT$(1)
40 IF Z$="s" THEN CALL FORMAT
50 IF Z$<>"n" THEN 30
```

CDBL

(F) CDBL (change double)

Converte um dado numérico no formato "precisão simples" em um dado no formato de "precisão dupla".

- FORMATO: CDBL (argumento)

- EXEMPLO: A#=CDBL(B!/2)

- FUNÇÃO: Esta função converte o argumento para o formato em precisão dupla. O valor resultante apresenta um formato com 17 dígitos. CDBL pode ser útil, quando se quer forçar uma operação a ser efetuada em precisão dupla, mesmo que os operandos sejam de precisão simples ou inteiros.

- PROGRAMA EXEMPLO:

```
10 REM PROGRAMA CDBL
20 A!=20
30 B=3.141592#
40 C=CDBL(A!/B)
50 PRINT A!,B,C
```

Fornece o caractere correspondente ao código especificado.

- **FORMATO:** CHR$ (argumento)
- **EXEMPLO:** A$=CHR$(65)
- **FUNÇÃO:** Realiza o inverso da função ASC, isto é, fornece o caractere correspondente ao código ASCII especificado. O argumento pode ser qualquer número de 0 a 255, ou qualquer expressão variável com um valor nesta faixa. O argumento ou expressão deve estar entre parênteses.
- **PROGRAMA EXEMPLO:**

```
10 ' TABLE DE CARACTERES
20 DATA 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
30 SCREEN 2,,,,1
40 OPEN"GRP:"FOR OUTPUT AS 1
50 FOR F=0 TO 255
60 L=F\16:C=4+F-16*L
70 PSET (11*C,13+11*L),1
80 IF F<32 THEN PRINT #1,CHR$(1)+CHR$(64+F)
90 IF F>31 THEN PRINT #1,CHR$(F)
100 NEXTF
110 FOR F=0 TO 187 STEP 11
120 LINE(31,F)-(218,F)
130 NEXT F
140 FOR F=31 TO 218 STEP 11
150 LINE(F,0)-(F,187)
160 NEXT F
170 FOR F=45 TO 217 STEP 11
180 C=C+1:READ A$
190 PSET(F+1,2),1:PRINT#1,A$
200 PSET(F+1,3),1:PRINT#1,A$
210 PSET(F,2),1:PRINT#1,A$
220 PSET(F,3),1:PRINT#1,A$
230 PSET(35,F-32),1:PRINT#1,A$
240 PSET(35,F-31),1:PRINT#1,A$
250 PSET(34,F-32),1:PRINT#1,A$
260 PSET(34,F-31),1:PRINT#1,A$
270 NEXT F
280 GOTO 280
```

CINT

(F) CINT (convert to interger)

Converte dados numéricos em números inteiros.

- **FORMATO:** CINT (argumento)
- **EXEMPLO:** A%=CINT(B#*2)
- **FUNÇÃO:** A função CINT (X) fornece o maior número inteiro possível menor que o valor do argumento especificado. Por exemplo:
  CINT (1.5) resulta em 1
  CINT (-1.5) resulta em -2.
  Para a função CINT, o argumento deve estar entre os limites de -32768 e +32767, caso contrário, ocasionará um erro de *overflow*. CINT pode ser utilizado para acelerar operações envolvendo operandos de precisão simples e dupla.
- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA CINT
20 CLS
30 FOR F=1 TO 20
40 A=500-RND(-TIME)*1000
50 PRINT A;"............";CINT(A)
60 NEXT F
```

CIRCLE (circle)

CIRCLE

Traça, no modo gráfico, um círculo, uma elipse, uma parte de um arco circular (ou um setor).

- **FORMATO:**

CIRCLE [STEP] (X,Y), raio [,cor] [,ângulo inicial] [,ângulo final] [,proporção]

(X,Y): Especifica as coordenadas que determinam o centro do círculo na tela. Podem ser constantes, variáveis ou variáveis indexadas, com valor entre -32768 e 32767. Se o STEP for especificado antes, a origem do sistema de coordenadas da tela (0,0) será transferida do canto superior esquerdo para a posição do último ponto *plotado.*

RAIO: Pode ser uma constante, variável ou uma variável indexada com valor entre -32768 a 32767, indicando o raio da figura a ser traçada.

COR: Indica qual a cor com que a elipse será desenhada. Deve ser um inteiro entre 0 e 15.

Os parâmetros [ângulo inicial] e [ângulo final], são medidos em radianos entre 0 e 2 PI (2π).

PROPORÇÃO: É a proporção entre o eixo horizontal e vertical da elipse. Os dados podem ser: constantes, variáveis indexadas, números positivos ou suas expressões. Se omitido, será assumido o valor 1

- **EXEMPLO:**

```
CIRCLE (128,86),70,10,0,3,4
```

- **FUNÇÃO:**

Este comando é utilizado para desenhar um círculo com a especificação do raio e com seu centro nas coordenadas também especificadas. Quando se especificam os ângulos (inicial e final), somente será traçado uma parte do arco circular. O arco poderá ser traçado colocando o sinal de menos ("-") para o valor dos ângulos inicial e final. Uma elipse poderá ser traçada especificando a "proporção" (relação altura/largura), isto é, o número de vezes que o eixo horizontal está contido na vertical.

- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA ESFERA
20 SCREEN2
30 FOR B=80 TO 1 STEP -10
40 E1=80/B
50 E2=B/80
60 CIRCLE (128,80),80,5,,,E1
70 CIRCLE (128,80),80,5,,,E2
80 NEXT B
90 COLOR 5,1
100 LINE (128,160)-(128,0)
110 LINE (48,80)-(208,80)
120 GOTO 100
```

CLEAR

CLEAR (clear)

Inicializa todas as variáveis e estabelece o tamanho da área de caracteres e o último endereço de memória utilizada pelo BASIC. Também fecha todos os arquivos abertos.

- **FORMATO:** CLEAR (tamanho da área de caracteres)[, (endereço da RAMTOP)]

  TAMANHO DA ÁREA DE CARACTERES: Espaço para variável *string*.

  ENDEREÇO DA RAMTOP: Especifica a área de memória disponível para ser usado pelo BASIC. Exemplo:

- **EXEMPLO:** CLEAR 400,55296

- **FUNÇÃO:** Com esta instrução, inicializa-se todas as variáveis e estabelece (quando especificado) o último endereço da área de programação BASIC, ou seja, o valor da RAMTOP.

- **PROGRAMA EXEMPLO:**

```
10 REM Programa CLEAR
20 CLEAR 200
30 ON ERROR GOTO 90
40 FOR F=1 TO 25
50 A$=A$+"##########"
60 PRINT"A$ esta com";10*F;"caracteres."
70 NEXT F
80 END
90 PRINT"Com";10*F;"caracteres em A$, o
micro nao"
100 PRINT"consegue trabalhar e ocorre um
 erro:"
110 PRINT CHR$(34);"Out of string space"
;CHR$(34)
120 PRINT:PRINT"Agora sera executado um
comando CLEAR    e o programa recomecara!
"
130 CLEAR 10000
140 FOR G=1 TO 3000 : NEXT G
150 GOTO 40
```

*CLOAD/CLOAD?

Carrega um programa em BASIC MSX de um cassete para a memória do MSX.

- **FORMATO:** CLOAD ["nome do arquivo"]
  CLOAD? ["nome do arquivo"]

- **EXEMPLO:**

```
CLOAD"DEMO"
CLOAD?"DEMO"
```

- **FUNÇÃO:** O comando CLOAD possibilita a carga de um programa armazenado em fita cassete (colocando o gravador no modo PLAY). Certifique-se de que as ligações foram feitas corretamente e de que a fita foi rebobinada até a posição correta. O nome de arquivo pode ter no máximo 6 caracteres entre aspas. Se ele possuir mais do que 6 caracteres, os excedentes serão desprezados. No nome de arquivo podem ser utilizados quaisquer caracteres, exceto as próprias aspas. O BASIC permite especificar o nome de arquivo desejado, por exemplo, CLOAD "EXPERT". Isso fará com que o computador ignore qualquer programa, até que ele encontre o programa rotulado "EXPERT". Quando o computador encontrá-lo, ele será carregado. Enquanto o computador estiver procurando o arquivo "EXPERT", os nomes dos outros programas encontrados aparecerão da seguinte forma:

```
Skip:GRADIE
Skip:ALEPH
Skip:MSX
```

e, quando o arquivo "EXPERT" for encontrado aparecerá:

```
Found:EXPERT
```

isto significa que "EXPERT" foi encontrado e está sendo carregado. Se o nome do arquivo não for especificado, o primeiro programa encontrado pelo computador será carregado na memória. CLOAD? permite a comparação de um programa armazenado em fita cassete, com um programa na memória do computador. Isto é útil quando, após gravar programas na fita (usando CSAVE), se deseja verificar se a transferência foi bem sucedida. Durante o CLOAD? o programa na fita e o programa na memória, são comparados byte por byte. Se houver alguma irregularidade (indicando uma gravação mal feita) o computador mandará a seguinte mensagem:

```
Verify error
```

Neste caso, você deverá gravar o programa novamente.

CLOSE

CLOSE (close)

Fecha um arquivo que foi aberto por um comando OPEN.

- **FORMATO:** CLOSE [#] [número do arquivo] [, número do arquivo]

- **EXEMPLO:** CLOSE#1

- **FUNÇÃO:** Tem a função de fechar o acesso a um arquivo, através de um ou mais buffers especificados, liberando-o(s). Se o número do arquivo não for especificado, todos os canais abertos pelo comando OPEN serão fechados.

- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA CLOSE
20 MAXFILES=1
30 CLS
40 OPEN "GRP:" FOR OUTPUT AS #1
50 SCREEN 2
55 LINE(0,0)-(255,191),5,BF
56 CIRCLE(120,120),50
60 PRINT #1,"EXPERT"
70 FOR I=1 TO 1000:NEXT
80 CLOSE 1
90 SCREEN 0
100 LIST
```

CLS (clear screen)

CLS

Apaga tudo o que está visualizado na tela.

- FORMATO: CLS

- EXEMPLO: CLS

- FUNÇÃO: Esse comando tem a função de "limpar a tela". Desativa todos seus pontos gráficos e move o cursor para o canto superior esquerdo. Muito últil quando se deseja uma boa apresentação visual. No modo gráfico o comando COLOR será executado após um comando CLS.

- PROGRAMA EXEMPLO:

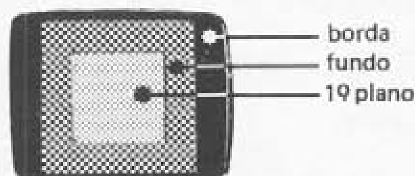
```
10 REM PROGRAMA CLS
20 INPUT"VOCE QUER LIMPAR A TELA (S/N)"
;R$
30 IF R$="N" THEN 50
40 CLS
50 END
```

COLOR

COLOR (color)

Especifica as cores do primeiro plano, fundo e área das bordas.

- FORMATO: COLOR [cor do 1º plano], [cor do fundo], [cor da borda]
- EXEMPLO: `COLOR 15,7,7`
- FUNÇÃO: Tem a função de definir a cor da tela que está dividida em:



Para definir a cor, o argumento deve estar entre 0 a 15. As cores correspondentes a cada valor estão na tabela abaixo:

cód.	cor	cód.	cor
0	transparente	8	vermelho
1	negro	9	vermelho claro
2	verde	10	amarelo escuro
3	verde claro	11	amarelo claro
4	azul escuro	12	verde escuro
5	azul claro	13	magenta
6	vermelho escuro	14	cinza
7	azul celeste	15	branco

Exemplo:

`COLOR 6` somente colocará a cor do primeiro plano

`COLOR ,2` somente colocará a cor do fundo

`COLOR ,,11` somente colocará a cor da área das bordas

`COLOR 15,1,1` cores de inicialização

- PROGRAMA EXEMPLO:

```
10 REM PROGRAMA COLOR
20 CLS
30 FOR F=1 TO 14
40 FOR G=0 TO150
50 NEXT G
60 PRINT"%%%%%%%% ALEPH    &  GRADIENTE
%%%%%%%%%"
70 COLORF,F-1,F+1
80 NEXTF
90 COLOR 15,1,1
```

CONT

Continua a execução de um programa interrompido.

- **FORMATO:** CONT

- **EXEMPLO:** CONT

- **FUNÇÃO:** Continua a execução de um programa interrompido por uma instrução END, STOP, ou mediante a digitação de CONTROL + STOP. O comando CONT reinicia a execução a partir da instrução seguinte àquela que foi interrompida, a não ser que a interrupção tenha ocorrido durante uma instrução INPUT. Nesse caso, CONT faz com que o INPUT seja executado desde o começo.

- **PROGRAMA EXEMPLO:**

```
10 ' CONT
20 FOR F=1 TO 20
30 PRINT TAB(F);F
40 NEXT F
50 PRINT"A execução do programa foi int
errompida por STOP."
60 PRINT"Digite CONT+RETURN para    con
tinuar!!!"
70 STOP
80 CLS
90 PRINT:PRINT"A execução esta continua
ndo!"
100 FOR R=333 TO 0 STEP-1
110 LOCATE 13,10
120 PRINTUSING"###";R
130 NEXT R
140 PRINT TAB(13);"FIM!!!"
150 END
```

COS

(F) COS (cosine)

Fornece o valor do cosseno de um arco em radianos.

- **FORMATO:** COS (argumento)

- **EXEMPLO:** C=COS(3.1415926535898/2)

- **FUNÇÃO:** A função COS (X) tem por finalidade fornecer o cosseno do argumento especificado que deve estar em radianos. A função COS é calculada em precisão dupla.

- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA COS
20 CLS
30 SCREEN 2
40 FOR F=0 TO 255 STEP .5
50 PSET(F,80-70*COS(F*6.141592#/255))
60 NEXT F
70 GOTO 70
```

CSAVE (cassete save)

*CSAVE

Armazena, num cassete, um programa em BASIC MSX.

- **FORMATO:** CSAVE "(nome do arquivo)" [, velocidade de transmissão em *bauds*]

NOME DO ARQUIVO: Adota no máximo 6 caracteres, sendo que o primeiro deve obrigatoriamente ser um caractere alfabético. Se forem especificados 7 ou mais caracteres, o sétimo e os demais serão ignorados.

VELOCIDADE DA TRANSMISSÃO EM *BAUDS*: Especifica a velocidade de gravação, adotando o seguinte critério: 1 para 1200 *bauds* e 2 para 2400 *bauds*. Se a velocidade for omitida, será adotada a velocidade 1 (=1200 *bauds*).

- **EXEMPLO:**

```
CSAVE"PROG1"
```

- **FUNÇÃO:** Gravar arquivos em fita cassete. A gravação é feita no formato binário condensado, de modo que não se pode usar o comando MERGE (ele só funciona para arquivos gravados em ASCII). A velocidade de transmissão de dados para o cassete também pode ser definida numa instrução SCREEN.

- **PROGRAMA EXEMPLO:**

```
10 CLS
20 PRINT"ESTE PROGRAMA SE AUTO-COPIA"
30 PRINT"APERTE (RETURN) PARA GRAVAR"
40 INPUT B$
50 CSAVE"ALEPH"
60 INPUT"OUTRA GRAVAÇÃO ";A$
70 IF A$="S" OR A$="s" THEN 10
80 END
```

CSNG

(F) CSNG (convert to single)

Converte um dado numérico para precisão simples.

- FORMATO: CSNG (argumento)
- EXEMPLO: `A!=CSNG(B#)`
- FUNÇÃO: Fornece uma representação em precisão simples do argumento. Quando o argumento for um valor de precisão dupla, esta função fornecerá um valor com seis dígitos significativos e um arredondamento de 4/5 no dígito menos significativo. Assim, temos:

  `CSNG(0.66666666666667)` resulta em `.666667`

  e

  `CSNG(0.33333333333333)` resulta em `.333333`

- PROGRAMA EXEMPLO:

```
10 REM PROGRAMA CSNG
20 A=12.15456723#
30 PRINT A,CSNG(A)
```

CSRLIN (cursor line)

CSRLIN

Fornece a linha em que se encontra o cursor.

- **FORMATO:** CSRLIN
- **EXEMPLO:** Y=CSRLIN
- **FUNÇÃO:** A função CSRLIN fornece o número da linha em que se encontra o cursor.
- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA CSRLIN
20 CLS
30 FOR F=1 TO 20
40 A$=A$+"#"
50 PRINT A$;CSRLIN
60 NEXT F
```

DATA

DATA (data)

Fornece os dados que serão lidos pelo comando READ.

- **FORMATO:** DATA constante [, constantes]

- **EXEMPLO:** `DATA 123,ABC,456`

- **FUNÇÃO:** A instrução DATA é utilizada para armazenar as constantes numéricas e as *strings* que serão acessadas pelo programa através da instrução READ. O comando DATA não é executável, ou seja, quando um programa que está sendo executado encontra-o, a instrução é desprezada e passa-se à execução da próxima instrução. Assim, um comando DATA pode ser colocado em qualquer parte do programa. Pode conter números e *strings* separados por uma vírgula. Um programa pode conter tantas instruções DATAs quantas forem necessárias. A instrução READ acessará os dados contidos nas instruções DATA na ordem em que foram definidas (por ordem do número da linha e pela seqüência dentro de uma instrução). As *strings* que possuam uma vírgula (,), dois-pontos (:) ou um espaço no começo ou no final, devem estar entre aspas (" "). As constantes contidas nas instruções DATA devem estar de acordo com a variável que a requisitará. O comando DATA pode ser lido novamente desde o começo ou a partir da linha especificada com o uso do comando RESTORE.

- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA DATA
20 DATA 0,1,3,5
30 DATA 3,4,1,0
40 DATA 4,6,0,9
50 FOR F= 1 TO 12
60 READ A
70 PRINT A
80 NEXT F
```

Define e atribui um nome a uma função escrita pelo usuário.

- **FORMATO:** DEF FN nome da função [(lista de parâmetros)] = (expressão da função definida)

- **EXEMPLO:**

```
DEF FN ARCCOTH(X)  = LOG((X+1)/(X-1))/2
```

- **FUNÇÃO:** A instrução DEF FN é utilizada para definir uma função que será usada num programa. Não pode ser utilizada como um comando direto. O nome da função a ser definida deve ser precedido pelo FN. A lista de parâmetros representa as variáveis que serão utilizadas pela função. Por exemplo:

```
10 DEF FNE(X,Y,Z)=X+7*Y/Z
```

Quando a função for chamada, cada parâmetro assumirá o valor que lhe for passado. Por exemplo, em:

```
20 A=FNE(4,16,8)
```

X = 4, Y = 16 e Z = 8 e a função FNE é executada e retornará com o valor em A igual a 18. A lista de parâmetros poderá conter no máximo 9 deles. A expressão da função a ser definida deve utilizar os parâmetros definidos. As variáveis do programa que contenham o mesmo nome dos parâmetros utilizados pela função não são afetadas. Porém pode-se utilizar na expressão da função uma variável que não foi definida na lista de parâmetros. Neste caso, quando a função for chamada, ela assumirá seu valor atual. Os valores passados para a função devem ser do mesmo tipo dos que foram especificadas na função. Se eles não coincidirem, uma mensagem de erro será enviada, indicando que o valor que foi passado para a função não está de acordo com o tipo definido na função. A expressão da função deve ser de apenas uma linha. A função deve ser definida antes de ser chamada. Caso contrário, será enviada uma mensagem de erro.

- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA DEF FN
20 PRINT" FORMULA DO DELTA"
30 DEF FN D(A)=B*B-4*A*C
40 INPUT"A=";A
50 INPUT"B=";B
60 INPUT"C=";C
70 PRINT"DELTA=";FN D(A)
```

DEFINT/SNG/DBL/STR

(define integer) (define single) (define double) (define string)

Define a correspondência entre o primeiro caractere de uma variável e o tipo de número que ela armazena.

- **FORMATO:**

```
DEFINT caractere [-caractere] [, caractere]
DEFSNG caractere [-caractere] [, caractere]
DEFDBL caractere [-caractere] [, caractere]
DEFSTR caractere [-caractere] [, caractere]
```

- **EXEMPLO:**

```
DEFINT A,I-K
```

- **FUNÇÃO:**

Este tipo de comando estabelece que os nomes das variáveis que começam com as letras especificadas deverão corresponder a um tipo específico de variável, ou seja:

DEFINT — define que as variáveis armazenarão números inteiros

DEFSNG — define que as variáveis armazenarão números de precisão simples

DEFDBL — define que as variáveis armazenarão números de precisão dupla

DEFSTR — define que as variáveis serão do tipo *string*.

O tipo de variável pode ser alterado durante um programa, desde que a prioridade quanto ao tipo de variável seja respeitada. A prioridade utilizada pelo MSX, é a seguinte:

$ — variável *string*
— variável de precisão dupla
! — variável de precisão simples
% — variável inteira

Por exemplo, uma variável definida como inteira pode ser alterada para uma variável de precisão *simples* ou dupla ou para uma variável *string*, mas uma variável *string* não pode ser alterada.

- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA DEFINT
20 DEFINT A,B,C
30 A=3.141592#
40 B=2.718281#
50 C=22.45525#
60 PRINT A,B,C
```

DEFUSR

Define um endereço para iniciar a execução de uma sub-rotina em linguagem de máquina que será chamada pelo comando USR.

- **FORMATO:** DEFUSR [dígito] = endereço para início de execução

- **EXEMPLO:** DEFUSR1=&HE00A

- **FUNÇÃO:** Este comando especifica o endereço inicial para a execução de uma sub-rotina em linguagem de máquina. A sub-rotina iniciará a execução quando ela for chamada pela instrução USR. O dígito a ser especificado deve ser um número inteiro entre 0 e 9 e corresponde ao número da rotina USR a ser executada. Se o valor do dígito for omitido, será assumido o valor 0. O endereço de início da rotina deve ser um número entre 0 a 65535. O endereço de início de sub-rotina pode ser redefinido posteriormente, ou seja, uma função DEF USR pode assumir valores diferentes durante a execução de um programa.

- **PROGRAMA EXEMPLO:**

```
10 DATA 3E,41,21,00,00,CD,4D,00,23,E5,D
5,11,BF,03,ED,52,D1,E1,38,F1,C9
20 FOR I=0 TO 20
30 READ A$:POKE 60000!+I,VAL("&H"+A$)
40 NEXT I
50 SCREEN 0:CLS:PRINT "APERTE <RETURN>
PARA EXECUTAR"
60 INPUT B$
70 DEF USR0=60000!
80 X=USR0 (0)
90 FOR T=1 TO 1000:NEXT T:CLS:END
```

DELETE

DELETE (delete)

Apaga as linhas de um programa.

- FORMATO: DELETE [número da linha A] [-número da linha B]

- EXEMPLO:

```
DELETE 10-230
DELETE 23
DELETE -120
```

- FUNÇÃO:

Este comando tem a função de apagar as linhas de um programa que está na memória. Pode-se especificar uma linha individual ou uma seqüência de linhas.

DELETE (linha A) — (linha B): apaga todas as linhas de programa começando pela linha A e terminando na linha B (inclusive).

DELETE (linha A): Apaga apenas a linha especificada.

DELETE -(linha B): Apaga desde a primeira linha do programa até a linha B (inclusive).

Você pode apagar a linha que acabou de introduzir, ou que acaba de ser evidenciada por uma mensagem de erro, digitando:

```
DELETE.
```

ou seja, ao invés de digitar um número de linha, digitar ponto (.).

- PROGRAMA EXEMPLO:

```
10 REM PROGRAMA DELETE
20 FOR F=0 TO 500
30 PRINT"ESTE PROGRAMA SE AUTO DESTROI"
40 NEXT F
50 DELETE -50
```

DIM (dimension)

DIM

Definição de uma ou mais variáveis do tipo matriz, especificando o tipo de matriz e a sua dimensão.

- FORMATO

DIM nome da variável (valor máximo de um sub-índice, ...)
[,nome da variável ...]

- EXEMPLO:

```
DIM Y(15,20)
DIM A$(2,3),C(4,8)
```

- FUNÇÃO:

Utilizada para declarar variáveis do tipo matriz, bem como o tipo de matriz que a variável será, além da sua dimensão. Se durante a execução de um programa for acessado um sub-índice maior do que o assumido, uma mensagem de erro será apresentada:

```
Subscript out of range
```

Numa sentença DIM podem ser declaradas mais de uma matriz, desde que separadas por uma vírgula.

VARIÁVEIS DE MATRIZ MULTIDIMENSIONAIS: As variáveis de matriz multidimensionais são geradas especificando-se dois ou mais valores máximos para o sub-índice. Por exemplo:

```
DIM A(3,4,5)
```

OMISSÃO DA SENTENÇA DIM: Quando se utilizar uma variável do tipo matriz sem ter declarado uma sentença DIM, o valor máximo do sub-índice será assumido como 10.

- PROGRAMA EXEMPLO:

```
10 REM PROGRAMA DIM
20 DIM I(12)
30 CLS:IA=1
40 FOR F= 1 TO 12
50 PRINT"INFLAÇÃO DO MES";F;":"
60 INPUT I(F)
70 IA=IA*(I(F)+100)/100
80 NEXT F
90 CLS
100 PRINT"INFLAÇÃO ANUAL:";IA*100-100
```

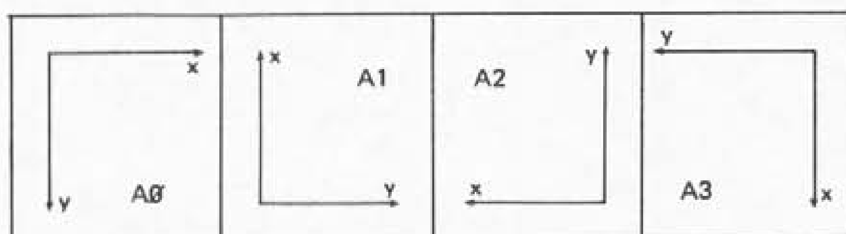
DRAW

Desenha figuras na tela gráfica (SCREEN 2 ou 3), de acordo com o sub-comando especificado.

- **FORMATO:** DRAW "sub-comandos"
- **EXEMPLO:** `DRAW "U20D10L30R40"`
- **FUNÇÃO:** A instrução DRAW desenha na tela de modo gráfico a figura especificada pelo sub-comando. São eles:

- **Sn (n entre Ø e 255):** Define a escala em que será traçada uma reta, ou seja, (1/4 pontos para n = 1). O valor inicial é S4.

- **An (n entre Ø e 3):** Gira o sistema de coordenadas de 90º em 90º, sendo que o valor inicial é A0.

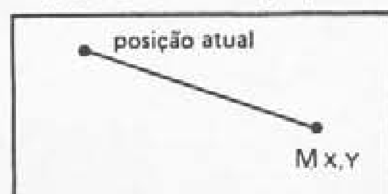


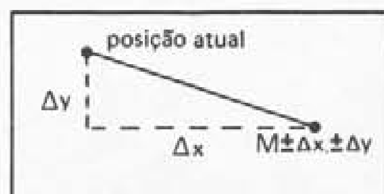
- **Cn (n entre Ø e 15):** Define a cor da linha que será traçada. Se omitido, assumirá C15 (15 = branco).

cód.	cor	cód.	cor
Ø	transparente	8	vermelho
1	negro	9	vermelho claro
2	verde	1Ø	amarelo escuro
3	verde claro	11	amarelo claro
4	azul escuro	12	verde escuro
5	azul claro	13	magenta
6	vermelho escuro	14	cinza
7	azul celeste	15	branco

- **M x, y (x entre Ø e 255; y entre Ø e 191):** Traça uma linha desde o último ponto plotado (posição atual) até a posição definida por x e y.

- **M ±Δx, ±Δy (Δx entre Ø e 255; Δ y entre Ø e 191):** Traça uma linha desde a posição atual até a posição em que a coordenada x é x atual ± Δ x e a coordenada y é y atual ± Δ y.



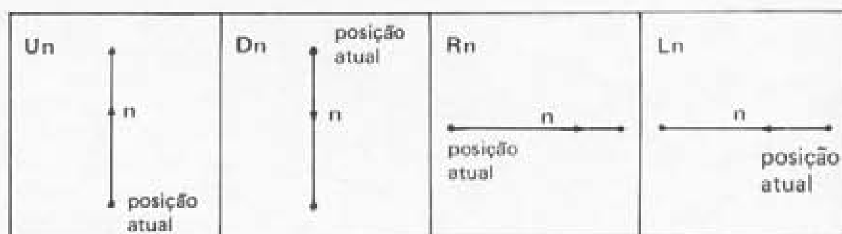


• **Un:** Traça uma linha vertical para cima desde a posição atual até uma distância n, onde n é o número de pontos definido pelo sub-comando S. Quando omitido assume o valor 1.

• **Dn:** Traça uma linha vertical para baixo desde a posição atual até uma distância n, onde n é o número de pontos definido pelo sub-comando S. Quando omitido assume o valor 1.

• **Rn:** Traça uma linha horizontal da esquerda para a direita desde a posição atual até uma distância n, onde n é o número de pontos definido pelo sub-comando S. Quando omitido assume o valor 1.

• **Ln:** Traça uma linha horizontal da direita para a esquerda desde a posição atual até uma distância n, onde n é o número de pontos definido pelo sub-comando S. Quando omitido assume o valor 1.

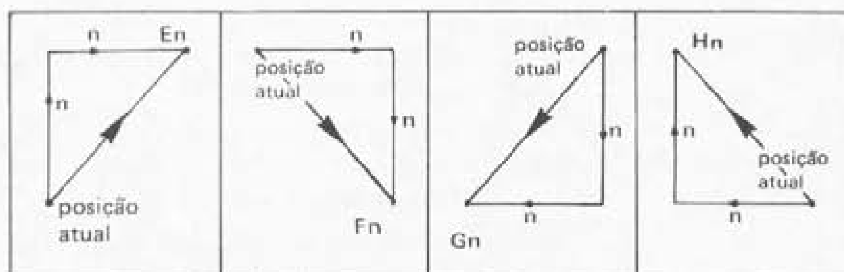


• **En:** É uma composição dos sub-comandos U e R, ou seja, traça uma reta desde a posição atual até a posição em que no eixo X a distância varie da esquerda para a direita de n posições e no eixo Y a distância varie de baixo para cima de n posições, onde n é o valor definido pelo sub-comando S. Quando omitido assume o valor 1.

• **Fn:** É uma composição dos sub-comandos D e R, ou seja, traça uma reta desde a posição atual até a posição em que no eixo X a distância varie da esquerda para a direita de n posições e no eixo Y a distância varie de cima para baixo de n posições, onde n é o valor definido pelo sub-comando S. Quando omitido assume o valor 1.

• **Gn:** É uma composição dos sub-comandos D e L, ou seja, traça uma reta desde a posição atual até a posição em que no eixo X a distância varie da direita para a esquerda de n posições e no eixo Y a distância varie de cima para baixo de n posições, onde n é o valor definido pelo sub-comando S. Quando omitido assume o valor a.

• **Hn:** É uma composição dos sub-comandos U e L, ou seja, traça uma reta desde a posição atual até a posição em que no eixo X a distância varie da direita para a esquerda de n posições e no eixo Y a distância varie de baixo para cima de n posições, onde n é o valor definido pelo sub-comando S. Quando omitido assume o valor 1.



A posição atual é armazenada após um dos sub-comandos exceto S, A e C. Antes dos sub-comandos que alteram o valor da posição atual pode-se utilizar os comandos:

- B: Posiciona o cursor na coordenada especificada sem que a reta seja traçada. Por exemplo:

```
BM 120,50
```

posiciona o cursor na posição de coordenadas (120,50).

- N: Traça a reta sem modificar o valor da posição atual. Por exemplo:

```
NU30
```

traça a reta, porém a posição do cursor não se modifica. O comando DRAW traça gráficos utilizando os sub-comandos citados anteriormente. Pode-se utilizar vários sub-comandos dentro de um mesmo comando DRAW, sendo que eles serão executados na ordem em que foram definidos. Por exemplo:

```
DRAW "BM20,40NU56R12"
```

Os sub-comandos podem também ser definidos em uma variável *string*. Por exemplo:

```
S$ = "BM80,50NG34F34"
```

e poderá ser executada especificando-se, depois do comando DRAW, a variável *string*. Por exemplo:

```
DRAW S$
```

Porém quando a variável *string* for utilizada juntamente com outros sub-comandos do comando DRAW, ou seja, entre as aspas (" ") ela deverá aparecer entre um X e um ponto e vírgula (;). Por exemplo:

```
DRAW "S6XS$;"
```

Quando uma variável numérica for utilizada dentro de um comando DRAW para expressar o valor do ângulo, a cor ou a distância, ela deverá ser precedida pelo sinal de igual (=) e sucedida por um ponto e vírgula (;). Por exemplo:

```
N=3
DRAW "BM45,60U=N;"
```

Observação: Antes do comando DRAW, deve-se executar a instrução:

```
SCREEN 2 ou SCREEN 3
```

que correspondem às telas gráficas.

Termina a execução de um programa, fecha todos os arquivos e retorna ao estado de espera de um comando direto.

- FORMATO: END
- EXEMPLO: END
- FUNÇÃO: Tem a função de finalizar a execução de um programa, fechando os arquivos abertos durante o programa e retornando ao estado de espera de um comando direto. É um comando diferente do comando STOP, pois o comando END não ocasiona uma mensagem de BREAK na tela. Quando as sub-rotinas são escritas logo abaixo do programa principal, costuma-se colocar um comando END como sendo a última linha do programa principal. Desta forma, evita-se que as sub-rotinas sejam executadas ao final da execução do programa principal. O comando END pode ser colocado em qualquer parte do programa e quantas vezes forem necessárias. Por exemplo, quando uma condição causa uma ramificação, pode-se colocar um comando END ao final de cada ramificação. O comando END no final do programa é opcional. Para continuar a execução de um programa que foi parado por um comando END, deve-se digitar RUN ou GOTO pois, com o comando CONT, não será possível o retorno à execução.
- PROGRAMA EXEMPLO:

```
10 ' END
20 CLS:PRINT"Agora, o programa principa
l esta sendo executado."
30 FOR F=333 TO 0 STEP -1
40 LOCATE 10,7:PRINT USING"###";F
50 NEXT F:CLS
60 GOSUB 90
70 PRINT"O comando END termina o progra
ma,      evitando que a sub-rotina da l
inha  90 seja executada novamente."
80 END
90 PRINT"Agora, a sub-rotina da linha 9
0 esta   sendo executada."
100 FOR F=0 TO 333
110 LOCATE 10,7:PRINT USING "###";F
120 NEXT F
130 RETURN
```

EOF *

(F) EOF (end of file)

Utilizado para testar o fim de um arquivo.

- FORMATO: EOF (número do arquivo)

- EXEMPLO:

```
IF EOF(1) THEN CLOSE#1:END
```

- FUNÇÃO: EOF é utilizado como uma variável para o controle de fim de arquivo. Quando um comando READ é executado e um registro foi lido do arquivo, EOF resulta em 0. Mas se o arquivo não possuía mais nenhum registro, EOF resulta em -1. EOF também é utilizado para testar o final de um arquivo e impedir erros do tipo:

```
Input past end
```

ou seja, entrada depois de um fim de arquivo.

- PROGRAMA EXEMPLO:

```
10 REM PROGRAMA EOF
20 OPEN"cas:II" FOR INPUT AS #1
30 IF EOF(1) THEN 70
40 INPUT #1,A$
50 GOTO 30
60 CLOSE #1
```

ERASE

Apaga, da memória, as variáveis do tipo matriz especificadas na instrução.

- **FORMATO:** ERASE nome da variável [, nome da variável ...]
- **EXEMPLO:** ERASE C,D
- **FUNÇÃO:** Libera a memória apenas na área utilizada pelas variáveis do tipo matriz especificadas na instrução ERASE. A área liberada pode ser utilizada para outros propósitos, ou então para um novo redimensionamento das variáveis apagadas. Se for feita uma tentativa de redimensionamento de uma variável que não foi apagada da memória, ocorrerá um erro do tipo:
  Redimensioned array
- **PROGRAMA EXEMPLO:**

```
10 DIM I(9)
20 FOR F=1 TO 9
30 READ I(F)
40 PRINT I(F)
50 NEXT F
60 ERASE I
70 FOR F=1 TO 9
80 PRINT I(F)
90 NEXT F
100 DATA 1,2,3,4,5,6,7,8,9
```

ERL/ERR

(F) ERL/ERR (error line/error)

Fornece o número da linha em que ocorreu um erro e o número do erro respectivamente.

- **FORMATO:**

```
ERL
ERR
```

- **EXEMPLO:**

```
L=ERL
E=ERR
```

- **FUNÇÃO:**

As variáveis ERL e ERR são geralmente utilizadas nas rotinas de erros de um programa com a instrução IF...THEN. Quando um erro é manuseado por um programa, a variável ERR fornece o código referente ao erro ocorrido, enquanto que a variável ERL fornece o número da linha em que o erro foi encontrado. Se for usado como um comando direto, ERL conterá 65535. Para testar se o erro ocorreu por um comando direto, digite:

```
IF 65535=ERL THEN ...
```

Caso contrário, comande:

```
IF ERL=(numero da linha) THEN ...
IF ERR=(codigo de erro) THEN ...
```

- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA ERR/ERL
20 ON ERROR GOTO 80
30 CLS
40 INPUT"DIGITE UM NUMERO NEGATIVO";A
50 PRINT SQR(A)
60 FOR F=1 TO 500: NEXT F
70 GOTO 30
80 PRINT"OCORREU O ERRO";ERR;"NA LINHA"
;ERL
90 PRINT"POIS NAO EXISTE RAIZ DE NUMERO
 NEGATIVO"
100 FOR F=1 TO 1000:NEXT F
110 RUN 20
```

ERROR (error)

ERROR

Simula a ocorrência de um erro e permite que o usuário defina um código de erro correspondente.

- FORMATO: ERROR número do erro
- EXEMPLO: ERROR 200
- FUNÇÃO: O comando ERROR simulará a ocorrência de um erro expondo a mensagem correspondente se o número do erro coincidir com um dos erros utilizados pelo BASIC MSX. Para definir seu próprio codigo de erro, utilize um valor maior do que qualquer valor usado pelo BASIC. Veja no apendice a lista de erros e suas respectivas mensagens. O número do erro deve estar entre 0 e 255, sendo que o BASIC MSX utiliza os códigos de 0 a 59. Estes codigos de erro podem depois ser convenientemente manuseados em uma rotina de erro. Por exemplo:

```
10 ON ERROR GOTO 1000
120 IF A$="S" THEN ERROR 250
1000 IF ERR=250 THEN PRINT "DESCULPE-ME"
```

Se um termo ERROR especificando um código não possui uma mensagem definida, o BASIC responderá com uma mensagem do tipo:

```
Unprintable error
```

que será exposta na tela e suspenderá a execução do programa.

- PROGRAMA EXEMPLO:

```
10 REM PROGRAMA ERROR
20 ON ERROR GOTO 80
30 PRINT"Digite a tecla A ."
40 A$=INKEY$
50 IF A$="" THEN 40
60 IF A$<>"A" AND A$<>"a" THEN ERROR 230
70 END
80 PRINT "A tecla A!!! (por favor)"
90 FOR F=1 TO 500:NEXT F
100 RUN
```

EXP

(F) EXP (exponential)

Fornece o valor de e^x, ou seja, da função exponencial natural de X.

- **FORMATO:** EXP (argumento)
- **EXEMPLO:** E=EXP(1)
- **FUNÇÃO:** Fornece o valor da exponencial natural do argumento especificado (e^X, onde e = 2.7182818284 . O valor do argumento pode ser menor ou igual a 145,06286085862. Se o valor do argumento for maior, uma mensagem do tipo:

  Overflow

  aparecerá na tela.
- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA EXP
20 CLS
30 SCREEN 2
40 FOR F=0 TO 255 STEP .5
50 PSET(F,191-290*EXP(-F/30))
60 NEXT F
70 GOTO 70
```

Fornece apenas a parte inteira de um dado numérico.

- FORMATO: FIX (argumento)

- EXEMPLO:

```
F=FIX(B/3)
```

- FUNÇÃO: A função FIX fornece apenas a parte inteira do argumento especificado, ou seja, fornece uma representação truncada de um dado numérico, sendo que todos os dígitos à direita do ponto decimal são desprezados. O argumento pode ser uma constante, uma variável numérica, uma variável numérica do tipo matriz ou então uma expressão matemática. A grande diferença entre as funções INT e FIX é que a função INT retorna com o maior inteiro possível, enquanto que a função FIX retorna com a parte inteira do argumento truncado, desprezando as casas decimais. Portanto, equivale a:

```
SGN(X)*INT(ABS(X))
```

pois esta diferença existe apenas para os números negativos.

- PROGRAMA EXEMPLO:

```
10 REM PROGRAMA FIX
20 CLS
30 FOR F=1 TO 20
40 A=500-RND(-TIME)*1000
50 PRINT A;"............";FIX(A)
60 NEXT F
```

FOR-NEXT

FOR-NEXT (for-next)

Repete a execução de um bloco de instruções entre um comando FOR e o seu correspondente NEXT.

- **FORMATO:** FOR variável = valor inicial TO valor final [STEP incremento]
  NEXT [variável]

- **EXEMPLO:**

```
FOR J=0 TO 100 STEP 2
NEXT J
```

- **FUNÇÃO:** O comando FOR-NEXT executa, repetidamente, um bloco de instruções contido entre as instruções FOR e NEXT correspondentes. A variável que controla o laço FOR-NEXT varia desde valor inicial até o valor final sendo acrescida pelo valor do incremento todas as vezes que o comando NEXT correspondente for executado. Quando o incremento não é definido, é adotado o valor 1. Por exemplo:

```
10 REM PROGRAMA FOR...NEXT...STEP...
20 CLS
30 FOR F=1 TO 40 STEP 2
40 G=G+F
50 PRINT F,G
60 NEXT F
```

apresentando na tela:

```
 1          1
 3          4
 5          9
 7          16
 9          25
 11         36
 13         49
 15         64
 17         81
 19         100
 21         121
 23         144
 25         169
 27         196
 29         225
 31         256
 33         289
 35         324
 37         361
 39         400
```

LAÇOS MÚLTIPLOS: Um loop FOR-NEXT pode ser colocado dentro de outro do mesmo tipo. Em tal caso, o laço interior deverá estar completamente incluído no laço externo e para cada laço deverão se utilizar variáveis diferentes. Por exemplo:

```
10 FOR I=1 TO 5 ─────────────┐
20 FOR J=1 TO I ─┐            │
30 PRINT"*";     │INTERIOR    │EXTERIOR
40 NEXT J ───────┘            │
50 PRINT                      │
60 NEXT I ────────────────────┘
```

```
*
**
***
****
*****
```

Com uma instrução NEXT pode-se terminar várias instruções FOR. Porém, não se pode omitir o nome das variáveis, que devem estar dispostos seqüencialmente, sendo que o laço mais interno deve estar em primeiro lugar e as variáveis devem estar separadas por vírgula. Por exemplo:

```
10 FOR I=0 TO 10
20 FOR J=0 TO 5
.
.
.
10010 NEXT J,I
```

- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA FOR...NEXT
20 FOR F=1 TO 20
30 G=G+F
40 PRINT F,G
50 NEXT F
```

FRE

FRE (free)

Fornece o número de bytes livres na RAM de uma área de memória não utilizada pelo BASIC MSX.

- **FORMATO:** FRE (X)
  FRE (" ")

- **EXEMPLO:**

```
FRE(0)
FRE("")
```

- **FUNÇÃO:** FRE (X) fornece o número de bytes na memória que podem ser usados por um programa BASIC, arquivos de texto, arquivos de um programa em linguagem de máquina, etc., ou seja, a área livre na memória. O valor de X pode ser um valor numérico qualquer.

  FRE (" ") fornece o número de bytes livre na área da memória utilizada pelas variáveis *strings.*

- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA FRE
20 CLS
30 PRINT"Existem";FRE(1);"bytes livres
e"
40 PRINTFRE("");"bytes para strings."
```

GOSUB-RETURN (go to subroutine – return) **GOSUB-RETURN**

Transfere a execução do programa para a sub-rotina especificada e indica o ponto de retorno ao programa principal.

- **FORMATO:** GOSUB número da linha A
  RETURN [número da linha B]

- **EXEMPLO:** GOSUB 1000

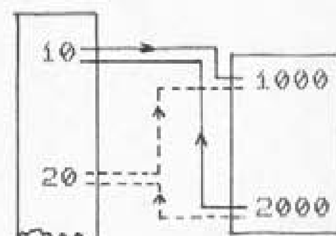
- **FUNÇÃO:** O comando GOSUB transfere a execução do programa para a sub-rotina que começa na linha A, enquanto que o comando RETURN indica o fim da sub-rotina e o retorno à linha B do programa principal. Quando o número de linha B é omitido, o retorno ao programa principal se dá na linha imediatamente seguinte à que contém o correspondente GOSUB. O número das linhas A e B devem estar entre 0 e 65535. Uma sub-rotina pode ser chamada quantas vezes forem necessárias. Por exemplo:

```
10 GOSUB 1000

20 GOSUB 1000

1000 (sub rotina)

2000 RETURN
```



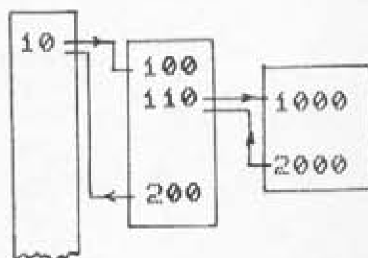
Uma sub-rotina pode chamar outra sub-rotina, porém o comportamento da multi-execução dependerá da capacidade da memória existente.

```
10 GOSUB 100

100 (sub rotina)


110 GOSUB 1000
200 RETURN
1000 (sub rotina)

2000 RETURN
```



- **PROGRAMA EXEMPLO:**

```
10 REM GOSUB ... RETURN
20 GOSUB 1000
30 GOSUB 2000
40 PRINT"ESTA ROTINA FAZ O MINIMO POSSI
VEL"
50 FOR F=1 TO 1000:NEXT F:PRINT
60 RUN
1000 PRINT"ESTA SUB-ROTINA SO IMPRIME"
1010 RETURN
2000 PRINT"ESTA SUB-ROTINA SO IMPRIME"
2010 RETURN
```

GOTO

GOTO (goto)

Desvia incondicionalmente a seqüência normal do programa para uma linha especificada.

- **FORMATO:** GOTO número da linha
- **EXEMPLO:** `GOTO 390`
- **FUNÇÃO:** A execução do programa é transferida para a linha especificada pelo comando GOTO. Se esta contiver uma instrução válida, será executada e, depois dela, as linhas seguintes. Se a linha indicada, porém, contiver uma instrução não executável (REM ou DATA), o micro a pulará passando para a primeira linha que contiver um comando válido, a partir da indicada pelo comando GOTO.

  Se a linha indicada pelo comando GOTO, contiver uma instrução mal-escrita, a tela exibirá a mensagem de erro:

  `Syntax error in (linha que contem o GOTO)`

  Se o número da linha indicada pelo GOTO não existir, será ativada a mensagem de erro:

  `Undefined line number in (linha com GOTO)`

  GOTO pode ser executado com comando direto.

- **PROGRAMA EXEMPLO:**

```
10 ' GOTO
20 PRINT"Linha 20":GOTO 50
30 PRINT"Linha 30"
40 PRINT"Linha 40":GOTO 60
50 PRINT"Linha 50":GOTO 30
60 PRINT"Linha 60"
70 PRINT"Linha 70":GOTO 90
80 PRINT"Linha 80":GOTO 70
90 PRINT" F I M"
```

HEX$ (hexadecimal dollar) (F) HEX$

Transforma um número em uma expressão hexadecimal na forma de uma *string*.

- FORMATO: HEX$ (X)
- EXEMPLO: PRINT HEX$(13)
- FUNÇÃO: A função HEX$ transforma um dado numérico em uma expressão hexadecimal na forma de uma *string*. X pode assumir um valor entre -32768 e 65535. Para X negativo, antes é calculado seu complemento (65536 - X) e a partir do resultado desta operação é feita a transformação, ou seja: HEX$ (-X) = HEX$ (65536-X)
- PROGRAMA EXEMPLO:

```
10 REM HEX$
20 CLS
30 PRINT" HEXADECIMAL  *****  ALEPH PUB
LICACOES"
40 PRINT
50 INPUT" QUANTOS NUMEROS";NS
60 PRINT :DIM H(NS)
70 CLS
80 PRINT" HEXADECIMAL  *****  ALEPH PUB
LICACOES"
90 FOR F=1 TO NS
100 INPUT"NUMERO";H(F)
110 NEXT F
120 CLS
130 PRINT" HEXADECIMAL  *****  ALEPH PU
BLICACOES"
140 FOR F=1 TO NS
150 PRINT"DECIMAL","HEXADECIMAL"
160 PRINT H(F),HEX$(H(F))
170 NEXT F
```

IF-THEN-ELSE

IF-THEN-ELSE (if-then-else)

Permite a bifurcação na execução do programa em função do cumprimento de uma condição.

- **FORMATO:** IF condição THEN comando se condição verdadeira [ELSE comando se condição falsa]

- **EXEMPLO:** IF A$="MSX" THEN GOTO 200 ELSE 300

- **FUNÇÃO:** Se a condição for verdadeira, será executado o comando após o termo THEN. Caso contrário, ou seja, se a condição for falsa, o comando após o termo THEN é ignorado e será executado o comando após o termo ELSE. Por exemplo:

X = 3 e Y = 8 IF X < Y ... (verdadeiro)
X = 9 e Y = 8 IF X < Y ... (falso)

O comando após o termo THEN pode inclusive ser um comando nulo, ou seja, não possuir nenhum comando. Se a condição for uma expressão entre parênteses, assumirá o valor −1 se verdadeira, e 0 se falsa. Por exemplo, na instrução:

```
IF (A$="MSX")=-1 THEN PRINT "OK"
```

será impresso um "OK", se a condição for verdadeira. No formato IF-THEN-GOTO pode-se omitir o termo THEN ou o termo GOTO. Por exemplo, se digitarmos:

```
IF A$="MSX" THEN 30     ou
IF A$="MSX" GOTO 30
```

obteremos o mesmo resultado, pois as duas expressões tem o mesmo significado. Porém, no formato IF-THEN-ELSE-GOTO, o termo ELSE pode ser seguido do comando GOTO número da linha ou simplesmente do número da linha (GOTO pode ser omitido, e mesmo assim, a execução será desviada, conforme o resultado da condição para o número da linha indicada). Depois do termo THEN e do termo ELSE, podem ser escritas várias sentenças. Elas serão executadas seqüencialmente, começando pela esquerda. Estas sentenças devem ser separadas por "dois pontos" (:).

COMANDO IF-THEN-ELSE ENCADEADOS: Depois de um termo THEN ou de um termo ELSE, pode ser também comandada uma nova condição. A este tipo de sentença dá-se o nome de comando encadeado.

- **PROGRAMA EXEMPLO:**

```
10 REM IF ... THEN ... ELSE
20 CLS : A=INT(10*RND(-TIME)) : PRINT
30 INPUT "Adivinhe o numero que pensei (
0-9).";N : BEEP
40 IF N<>A THEN GOTO 80 ELSE 50
50 PRINT "Muito bem, voce acertou !!!"
60 FOR F=1 TO 1000 : NEXT F
70 GOTO 20
80 PRINT "Voce errou! Tente outra vez!"
90 GOTO 30
```

INKEY$

Obtém o caractere da tecla pressionada.

- FORMATO: INKEY$
- EXEMPLO: C$=INKEY$
- FUNÇÃO: A função INKEY$ obtém o caractere da tecla (exceto CTRL + STOP, SHIFT e CTRL) que está sendo pressionada. Se nenhuma tecla estiver pressionada, o resultado será um caractere nulo.
- PROGRAMA EXEMPLO:

```
10 REM INKEY$
20 X=128
30 Y=96
40 SCREEN 2
50 PSET(128,96),10
60 DRAW"E2F2G2H2"
70 X=X-(INKEY$="D")+(INKEY$="A")
80 Y=Y-(INKEY$="X")+(INKEY$="W")
90 PSET(X,Y),10
100 GOTO 60
```

INP *

(F) INP (input)

Lê os dados da via de acesso especificada.

- FORMATO: INP (número da via de acesso)

- EXEMPLO:

```
A=INP(15)
```

- FUNÇÃO: Este comando lê dados através da via de acesso especificada pelo número entre parênteses. Este número pode ser fornecido através de uma constante, uma variável numérica, uma variável do tipo matriz ou uma expressão matemática e o seu valor deve estar compreendido entre 0 e 255.

- PROGRAMA EXEMPLO:

```
10 CLS:PRINT "EXPERIMENTE APERTAR <CAPS
 LOCK> !"
20 LOCATE 1(,10:X=INP(170):IF X=26 THEN
 PRINT "MAIUSCULAS" ELSE PRINT "minuscu
las"
30 GOTO 20
```

INPUT

Introduz o valor de uma variável através do teclado.

- **FORMATO:** INPUT ["mensagem";] variável [, variável] . . .

- **EXEMPLO:** INPUT"QUAL O SEU NOME";N$

- **FUNÇÃO:** Quando um comando INPUT é encontrado no programa, um ponto de interrogação é colocado na tela e a execução do programa é interrompida à espera de que um dado seja fornecido pelo usuário via teclado. Quando a mensagem é especificada, ela é impressa na tela, seguida do ponto de interrogação. Deve-se digitar o dado requisitado que será transferido para a variável ou variáveis na ordem em que foram fornecidos ao MSX. O número de dados a serem introduzidos deve ser igual ao número de variáveis, sendo que cada dado deve estar separado por uma vírgula. Se for digitado um número maior do que o necessário, aparecerá na tela uma mensagem do tipo:

```
Extra ignored
```

indicando que os dados excedentes foram ignorados e se passa à execução do próximo comando. Porém, se o número de dados introduzidos for menor do que o requisitado, aparecerão na tela dois pontos de interrogação (??) indicando que o MSX está à espera de mais dados. O tipo de dado introduzido deve estar de acordo com o tipo de variável especificado, caso contrário, uma mensagem do tipo:

```
Redo from start
```

aparecerá na tela, e o comando INPUT será executado novamente. Digitando CTRL + STOP ou CTRL + C, o MSX retorna ao nível de comando direto mandando uma mensagem do tipo:

```
Break in (numero da linha do comando INPUT)
```

Para continuar a execução, basta comandar CONT e o programa retornará à execução do comando INPUT.

- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA INPUT
20 PRINT "TUDO O QUE FOR ESCRITO DEPOIS
 DO PONTO DE INTERROGACAO, SERA ESCRITO
 NA TELA!"
30 INPUT"QUAL O SEU NOME";A$
40 PRINT A$
50 END
```

INPUT

INPUT # (input number)

Lê dados de um arquivo aberto, associando-os às variáveis de um programa.

- FORMATO: INPUT # número do arquivo, lista de variáveis

- EXEMPLO:

```
INPUT#1,A,B$
```

- FUNÇÃO: A instrução INPUT# lê dados de um arquivo que foi aberto por uma instrução OPEN e associa os dados às variáveis listadas. O número do arquivo deve ser um número maior do que zero e menor ou igual ao número de arquivos especificado em MAXFILES. A lista de variáveis inclui as variáveis numéricas, alfanuméricas (*strings*) e as indexadas.

- PROGRAMA EXEMPLO:

```
10 REM PROGRAMA INPUT#
20 OPEN "CAS:II" FOR INPUT AS #1
30 IF EOF(1)=-1 THEN 70
40 INPUT #1,A$
50 PRINT A$:BEEP
60 GOTO 30
70 CLOSE #1
```

INPUT$ (input dollar)

INPUT$

Lê um número especificado de caracteres introduzidos pelo teclado ou através de um arquivo.

- **FORMATO:** INPUT$ (número de caracteres)
  INPUT$ (número de caracteres,
  [, # número do arquivo]])

- **EXEMPLO:**

```
X$=INPUT$(6)
W$=INPUT$(3,#1)
```

- **FUNÇÃO:** O formato INPUT$ (número de caracteres) lê, do teclado o número especificado de caracteres e os fornece na forma de uma *string*. O formato INPUT$ (número de caracteres, # número do arquivo) lê, de um arquivo que está aberto, o número de caracteres especificado, e os fornece na forma de uma *string*. O número de caracteres pode ser fornecido por uma constante, uma variável numérica ou então por uma variável numérica indexada. O valor do número de caracteres deve estar entre 1 e 255 (se for maior do que 200, a instrução CLEAR deve ser executada antes).

- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA INPUT$
20 A$=INPUT$(8)
30 PRINT A$
```

INSTR

(F) INSTR (in *string*)

Localiza a posição de uma *string* dentro de uma outra *string*.

- **FORMATO:** INSTR ([N,] *string* de pesquisa, *string* procurada)

- **EXEMPLO:** `INSTR("SOU O MSX","O")`

- **FUNÇÃO:** A função INSTR fornece a localização do primeiro caractere da *string* procurada dentro da *string* de pesquisa, sendo que esta contagem começa da esquerda para a direita. Quando N é especificado, a procura iniciar-se-á a partir do n-ésimo caractere da *string* de pesquisa. N pode ser fornecido através de uma constante, uma variável numérica ou uma variável numérica indexada, sendo que seu valor deve estar entre 1 e 255. As *strings* de pesquisa e de procura podem ser fornecidas por uma constante, variável alfanumérica ou uma variável alfanumérica indexada. Se o valor N for maior do que o comprimento da *string* de pesquisa, ou se a *string* de pesquisa for uma *string* nula, ou se a *string* procurada não for encontrada na *string* de pesquisa, o resultado será zero (0).

- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA INSTR
20 A$="GRADIENTE&ALEPH-HPELA&ETNEIDARG"
30 INPUT"ENTRE UMA SEQUENCIA DE ATE 10
         CARACTERES";B$
40 C=INSTR(A$,B$)
50 IF C=0 THEN GOTO 80
60 PRINT"ESSA SEQUENCIA COMECA NA POSIÇ
ÃO";C
70 END
80 PRINT"NÃO EXISTE ESSA SEQUENCIA EM A
$"
90 END
```

INT (integer) (F) **INT**

Fornece o valor do maior inteiro possível, menor do que o argumento.

- **FORMATO:** INT (argumento)
- **EXEMPLO:** PRINT INT(1.12345)
- **FUNÇÃO:** A função INT fornece o valor do maior inteiro possível que não seja maior do que o valor do argumento. Por exemplo:

```
INT(2.5653)=2
INT(-2.5653)=-3
INT(1000101.23)=1000101
```

O argumento pode ser fornecido por uma constante, uma variável numérica ou uma variável numérica indexada.

- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA INT
20 CLS
30 FOR F=1 TO 20
40 A=500-RND(-TIME)*1000
50 PRINT A;"............";INT(A)
60 NEXT F
```

INTERVAL ON/OFF/STOP (interval on/off/stop)

Cada uma destas três funções respectivamente, habilita, desabilita e adia uma interrupção feita pelo temporizador.

- FORMATO:

```
INTERVAL ON
INTERVAL OFF
INTERVAL STOP
```

- EXEMPLO:

```
INTERVAL ON
INTERVAL OFF
INTERVAL STOP
```

- FUNÇÃO:

A instrução INTERVAL ON habilita uma interrupção causada pelo temporizador, caso exista uma instrução ON INTERVAL GOSUB no programa. A instrução INTERVAL OFF desabilita (ou desliga) a instrução INTERVAL ON. A instrução INTERVAL STOP adia a interrupção até que seja encontrada a instrução INTERVAL ON.

- PROGRAMA EXEMPLO:

```
10 REM PROGRAMA INTERVAL ON
20 INTERVAL ON
30 ON INTERVAL=50 GOSUB 50
40 GOTO 40
50 C=C+1
60 PRINT C
70 RETURN
```

KEY (key)

KEY

Associa uma *string* (cadeia de caracteres) a uma tecla de função (F1 a F10).

- FORMATO: KEY número da tecla de função, *string*

- EXEMPLO:

```
KEY 1,"LOAD"
```

- FUNÇÃO: A função KEY associa uma *string* a uma tecla de função, localizada na parte superior esquerda do teclado. O número da tecla de função deve estar entre 1 e 10 (as teclas de F6 a F10 são obtidas mantendo-se a tecla SHIFT pressionada) e a *string* deve ter, no máximo, 15 caracteres. Esta associação será desfeita se o computador for desligado. Podemos também utilizar a função CHR$ na formação da cadeia de caracteres (por exemplo, o caractere de retorno, que não possui um correspondente alfanumérico, entra como código 13). Exemplos:

```
KEY 1,"gradiente"+"aleph"

KEY 2,"LIST"+CHR$(13)

KEY 7,"FOR I="

KEY 8,"NEXT I"
```

- PROGRAMA:
  EXEMPLO:

```
10 ' KEY
20 KEY 1,"CLS:LIST"+CHR$(13)
30 INPUT"Voce quer definir alguma outra
 tecla";R$
35 IF R$="n" THEN END
40 INPUT"Qual a tecla (1-10)";T
50 INPUT"Qual a funcao";F$
60 KEY T,F$
70 GOTO 30
```

KEY LIST

Visualiza o conteúdo das teclas de função.

- FORMATO: KEY LIST

- EXEMPLO:

```
KEY LIST
```

- FUNÇÃO: O comando KEY LIST, permite a visualização do conjunto de caracteres associado às teclas de função. O estado inicial (default) que o computador oferece é o seguinte:

```
F1 . . . . . . . . . . . . . . . . . . . "color"  + CHR$ (32)
F2 . . . . . . . . . . . . . . . . . . . "auto"   + CHR$ (32)
F3 . . . . . . . . . . . . . . . . . . . "goto"   + CHR$ (32)
F4 . . . . . . . . . . . . . . . . . . . "list"   + CHR$ (32)
F5 . . . . . . . . . . . . . . . . . . . "run"    + CHR$ (13)
F6 (F1 + SHIFT) . . . . . . . . "color"  + CHR$ (32) +
                                "15,1,1" + CHR$ (13)
F7 (F2 + SHIFT) . . . . . . . . "cload"  + CHR$ (34)
F8 (F3 + SHIFT) . . . . . . . . "cont"   + CHR$ (13)
F9 (F4 + SHIFT) . . . . . . . . "list"   + CHR$ (13)
F10 (F5 + SHIFT) . . . . . . .  "run"    + CHR$ (13)
```

Observação: CHR$ (13) = RETURN
CHR$ (32) = ESPAÇO
CHR$ (34) = ASPAS (")

- PROGRAMA EXEMPLO:

```
10 REM PROGRAMA KEY LIST
20 INPUT"VOCE QUER VERIFICAR AS TECLAS
FUNCIONAIS (S/N)";R$
30 IF R$="N" THEN 50
40 KEY LIST
50 END
```

KEY ON/OFF

Ativa ou desativa a visualização, na parte inferior da tela, dos caracteres associados às teclas de função.

- **FORMATO:** KEY ON
  KEY OFF

- **EXEMPLO:**

```
KEY ON
KEY OFF
```

- **FUNÇÃO:** O comando KEY ON ativa a visualização na parte inferior da tela da cadeia de caracteres associada a cada tecla de função. O comando KEY OFF desativa esta visualização.

- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA KEY ON/OFF
20 INPUT"VOCE QUER AS TECLAS FUNCIONAIS
 NA TELA (S/N)";R$
30 IF R$="N" THEN KEY OFF ELSE KEY ON
40 END
```

KEY (n) ON/OFF/STOP

Cada uma destas três funções respectivamente, habilita, desabilita e adia uma interrupção através das teclas de função.

- **FORMATO:**

  KEY (n) ON
  KEY (n) OFF
  KEY (n) STOP

- **EXEMPLO:**

```
KEY(1) ON
KEY(2) OFF
KEY(3) STOP
```

- **FUNÇÃO:**

  A instrução KEY (n) ON habilita uma interrupção causada caso seja pressionada uma das teclas de função, quando existir a instrução ON KEY GOSUB no programa. A instrução KEY (n) OFF desabilita (ou desliga) a função KEY (n) ON. A instrução KEY (n) STOP adia a interrupção até que seja encontrada a instrução KEY (n) ON.

- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA KEY(n) ON
20 KEY(1) ON
30 ON KEY GOSUB 100
40 GOTO 30
100 PRINT"VOCE PRESSIONOU A TECLA F1."
110 RETURN
```

LEFT$ (left dollar) (F)

LEFT$

Fornece na forma de *string* os n-primeiros caracteres de uma cadeia.

- FORMATO: LEFT$ (*string*, quantidade de caracteres)

- EXEMPLO: LEFT$("GRADIENTE-ALEPH",5)

- FUNÇÃO: A função LEFT$ fornece os n-primeiros caracteres de uma *string*, sendo o primeiro caractere o mais à esquerda da *string*.

- PROGRAMA EXEMPLO:

```
10 REM PROGRAMA LEFT$
20 CLS
30 INPUT " ESCREVA UMA PALAVRA";W$: PRI
NT
40 PRINT "A PRIMEIRA LETRA E'...";LEFT$
(W$,1): PRINT
```

LEN

(F) LEN (lenght)

Fornece a quantidade de caracteres de uma *string*.

- **FORMATO:** LEN (*string*)
- **EXEMPLO:** LEN("COMPUTADOR")
- **FUNÇÃO:** A função LEN fornece a quantidade de caracteres de uma *string*, ou seja, o seu comprimento.
- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA LEN
20 CLS
30 INPUT"ESCREVA UMA SENTENCA QUE NAO U
LTRAPASSE 10 PALAVRAS.";S$: PRINT
40 PRINT " O NUMERO DE CARACTERES QUE E
STA FRASE POSSUI E':"
50 PRINT LEN(S$)
```

LET

Realiza a associação de um dado a uma variável.

- **FORMATO:** [LET] variável = expressão

- **EXEMPLO:**

```
LET A=10
LET A="ABC"+"DEF"
```

- **FUNÇÃO:** A instrução LET faz a associação de um dado (numérico ou alfanumérico) com uma variável (do mesmo tipo do dado), sendo que as variáveis alfanuméricas devem estar entre aspas (" "), caso contrário, uma mensagem do tipo:

```
Type mismatch
```

aparecerá na tela. A palavra LET pode ser omitida neste tipo de instrução.

- **PROGRAMA: EXEMPLO:**

```
10 REM PROGRAMA LET
20 CLS
30 LET A=10:PRINT"A=";A
40 LET B=20:PRINT"B=";B
50 LET C=A+B
60 PRINT "A+B=";C
70 END
```

LINE

LINE (line)

Traça, no modo gráfico, uma linha ou um quadrado, dependendo da sintaxe da instrução.

- **FORMATO:**

```
LINE [[STEP] (X1,Y1) ] -
      [STEP] (X2, Y2)
      [cor] [,B] [,BF]
```

- **EXEMPLO:**

```
LINE(100,100)-(135,145),3,B
```

- **FUNÇÃO:**

O comando LINE traça uma linha reta entre os pontos de coordenadas (X1, Y1) e (X2, Y2). Esta reta poderá ser colorida, se o ítem cor for especificado. Quando a letra B (box) é especificada, será traçado um retângulo que ligará "diagonalmente" os dois pontos especificados. Se BF (box full) for especificado, será traçado um retângulo com a parte interna pintada pela cor especificada. Quando a cor não for especificada, será usada a cor do primeiro plano definida pela última instrução COLOR.
Por exemplo:

```
LINE(100,130)-(120,150),,B
```

Para desenhar um quadrado, a diferença entre X1 e X2 deve ser 1,25 vezes menor do que a diferença entre Y1 e Y2. Isto porque a relação entre os caracteres horizontais e verticais é de 8:10. Por exemplo, em:

```
LINE(50,50)-(130,150)
```

```
X1-X2=-80, Y1-Y2=-100 e (Y1-Y2)/(X1-X2)=1.25
```

Se esta relação entre o eixo X e o eixo Y não for obedecida, será desenhado um retângulo. Por exemplo:

```
LINE(10,70)-(40,80),3,B
```

Observação: Antes de uma instrução LINE, deve existir um comando SCREEN 2 ou SCREEN 3, que muda para a página gráfica.

- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA LINE
20 SCREEN2
30 FOR F=0 TO 125  STEP 10
40 G=80-F*80/125
50 LINE (F,80)-(125,G)
60 LINE (125,G)-(250-F,80)
70 LINE (F,80)-(125,160-G)
80 LINE (125,160-G)-(250-F,80)
90 NEXT F
100 GOTO 100
```

LINE INPUT

Associa um dado introduzido pelo teclado, a uma variável *string*.

- **FORMATO:** LINE INPUT ["mensagem";] variável

- **EXEMPLO:** `LINE INPUT"Entre com um nome:";A$`

- **FUNÇÃO:** A instrução LINE INPUT executa a leitura do teclado e associa os caracteres pressionados a uma variável *string*. É uma instrução semelhante ao INPUT, com a diferença de que o LINE INPUT não coloca o ponto de interrogação na tela e associa os caracteres digitados a apenas uma variável. A mensagem é opcional e, quando especificada, será mostrada na tela.

- **PROGRAMA EXEMPLO:**

```
10 ' LINE INPUT
20 LINE INPUT"Introduza o seu primeiro
 nome e digite RETURN.     ";N$
25 CLS:N$=N$+"        "
30 N$=RIGHT$(N$,LEN(N$)-1)+LEFT$(N$,1):
LOCATE 10,10
40 PRINT N$:FOR F=1 TO 30:NEXT F
50 GOTO 30
```

*LINE INPUT

LINE INPUT # (line input number)

Lê uma seqüência de até 254 caracteres de um arquivo e a atribui a uma variável *string*.

- FORMATO: LINE INPUT # número do arquivo,nome da variável

- EXEMPLO: `LINE INPUT#2,A$`

- FUNÇÃO: O comando LINE INPUT # obtém uma *string* de um arquivo e a armazena numa variável. O arquivo é definido por um número entre 1 e o número especificado por MAXFILES, e a variável pode ser simples ou indexada.

- PROGRAMA EXEMPLO:

```
10 REM PROGRAMA LINE INPUT
20 OPEN "CAS:II" FOR INPUT AS#1
30 IF EOF=-1 THEN 70
40 LINE INPUT#1,A$
50 PRINT A$:BEEP
60 GOTO 30
70 CLOSE#1
```

LIST/LLIST (list out/line printer list out)

LIST/ LLIST

Lista o programa ou parte dele na tela ou na impressora.

- **FORMATO:**

LIST [número da linha inicial]
[-número da linha final]

LLIST [número da linha inicial]
[-número da linha final]

- **EXEMPLO:**

```
LIST-900
LIST 30-70
```

- **FUNÇÃO:**

Os dois comandos (LIST e LLIST) listam todo ou parte de um programa sendo que com o comando LIST a listagem do programa é visualizada na tela, enquanto que o comando LLIST é executado na impressora. Os números das linhas inicial e final são opcionais e, portanto, podem ser omitidos. Se for feito um comando LIST sem os números de linhas, todo o programa será listado. Se for feito um comando com apenas um número de linha, será listada apenas a linha especificada. Por exemplo:

```
LIST 50
```

Outra opção seria um comando com um hífen antes do número da linha. Neste caso, o programa será listado do início até a linha especificada. Por exemplo:

```
LIST-900
```

- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA LIST/LLIST
20 INPUT"VOCE QUER LISTAR O PROGRAMA NA
 TELA (T)OU NA IMPRESSORA (I)";R$
30 IF R$="T" THEN LIST
40 IF R$="I" THEN LLIST
50 END
```

LOAD*

LOAD (load)

Carrega um arquivo de um dispositivo especificado para a memória do micro.

- FORMATO: LOAD "[nome do dispositivo] [nome do arquivo]" [,R]
- EXEMPLO: `LOAD"CAS:MARRE",R`
- FUNÇÃO: Carregar, para a memória do computador, um arquivo em BASIC gravado no dispositivo especificado. Se a letra [R] for colocado após a última aspas o programa será executado assim que a transferência para a RAM for completada.

LOCATE

Move o cursor para posição especificada pelas coordenadas x e y.

- **FORMATO:** LOCATE [coordenada x] [, coordenada y] [, interruptor do cursor]

- **EXEMPLO:**

```
LOCATE 10,10,1
```

- **FUNÇÃO:** A instrução LOCATE move o cursor para uma posição especificada pelas coordenadas X e Y.

Coordenada X: constantes, variáveis indexadas ou numéricas, podendo assumir valores entre 0 e 39. Se omitida, o computador assumirá o valor 0.

Coordenada Y: constantes, variáveis indexadas ou numéricas, podendo assumir valores entre 0 e 23. Se omitida, o computador assumirá o valor 0.

Interruptor do Cursor: se for 0 o cursor não será visível. Se for 1, o cursor aparecerá. Se omitido, o computador assumirá o valor 1.

- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA LOCATE
20 INPUT"INTRODUZA UMA PALAVRA";P$
30 INPUT"EM QUE COLUNA VOCE QUER IMPRIM
I-LA";CO
40 INPUT"E EM QUE LINHA";LI
50 CLS
60 LOCATE CO,LI
70 PRINT P$
```

LOG

(F) LOG (logaritmo natural)

Determina o logaritmo natural de um número.

- **FORMATO:** LOG (X)
- **EXEMPLO:** LOG(10)
- **FUNÇÃO:** A função LOG determina o logaritmo natural cuja base é .... (2.7182818284588...). Para determinar o valor de um logaritmo em outra base (A):

  $LOG_A B$

  sendo B > 0 e A (A = base) um número positivo diferente de zero, utiliza-se a fórmula:

  LOG (B)/LOG (A)

  Observação: o valor de X sempre deverá ser maior que zero.

- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA LOG
20 INPUT"INTRODUZA UM NUMERO POSITIVO:"
;N
30 PRINT"O LOGARITMO NATURAL DESSE NUME
RO E:";LOG(N),"E SEU LOGARITMO DECIMAL
E:";LOG(N)/LOG(10)
40 PRINT
50 RUN
```

LPOS (line printer position) (F)

LPOS

Fornece a posição do cabeçote da impressora.

- **FORMATO:** LPOS (X)
- **EXEMPLO:** LPOS(0)
- **FUNÇÃO:** Fornece a posição no *buffer* de memória da impressora, do caractere que está sendo impresso (posição inicial = 0). O valor de X pode ser um número arbitrário qualquer e o valor obtido é um número inteiro.
- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA LPOS
20 A$=""
30 FOR F=1 TO 60
40 LPRINT A$;LPOS(1)
50 A$=A$+">"
60 NEXT F
```

LPRINT

LPRINT (line print)

Escreve na impressora o valor de uma expressão.

- FORMATO: LPRINT expressão [expressão...]
- EXEMPLO: LPRINT A$,B;"ALEPH",D+2
- FUNÇÃO: Escrever dados na impressora, da mesma forma que o comando PRINT os escreve na tela. As expressões podem ser: constantes, qualquer tipo de variável ou expressões algébricas. Entre as expressões podem ser utilizados os símbolos "," ou ";". O comando LPRINT desacompanhado de qualquer expressão ou símbolo causa o avanço de uma linha. Para maiores detalhes consulte PRINT.
- PROGRAMA EXEMPLO:

```
10 REM PROGRAMA LPRINT
20 FOR F=1 TO 50
30 LPRINT
40 LPRINT
50 LPRINT
60 FOR G=1 TO 60
70 LPRINT"------------------------------------------------------------------------------------------------------------"
80 NEXT G
90 LPRINT
100 LPRINT
110 LPRINT TAB 70;F
120 NEXT F
```

LPRINT USING (line print using)

LPRINT USING

Escreve na impressora, com o formato especificado, o valor de uma expressão.

- **FORMATO:** LPRINT USING "símbolo de formato" [expressão]

- **EXEMPLO:**

```
LPRINT USING"####.##";A$;B
```

- **FUNÇÃO:** Escreve dados na impressora, com formato determinado, da mesma forma que o comando PRINT USING os escreve na tela. As expressões podem ser: constantes, qualquer tipo de variável ou expressões algébricas. Entre as expressões podem ser utilizados os símbolos "," ou ";". Para maiores detalhes consulte PRINT USING.

- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA LPRINT USING
20 INPUT A
30 LPRINT USING"$$######.##-";A
40 GOTO 20
```

MAXFILES *

MAXFILES (maxfiles)

Determina o número de arquivos que podem ser abertos ao mesmo tempo em um programa.

- **FORMATO:** MAXFILES = número de arquivos

- **EXEMPLO:** MAXFILES=5

- **FUNÇÃO:** Declara o número de arquivos que podem ser abertos ao mesmo tempo em um programa. O número de arquivos pode assumir valores entre 0 e 15. Se o número de arquivos não for especificado através de MAXFILES, somente um pode ser aberto. Não se deve superdimensionar o valor de MAXFILES (quanto maior for, menor a área de memória restante para o usuário). O número de arquivos pode ser constituído de constantes, variáveis numéricas, variáveis indexadas, ou suas expressões.

- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA MAXFILES
20 MAXFILES=2
30 OPEN"GRP:"FOR OUTPUT AS #1
40 OPEN"CRT:"FOR OUTPUT AS #2
50 SCREEN 2
60 LINE(20,20)-(235,171),7,BF
70 LINE(50,50)-(205,141),4,BF
80 CIRCLE(128,80),30,6
90 PSET(107,70)
100 PRINT#1,"EXPERT"
110 CIRCLE(128,100),20,6
120 PSET(119,94)
130 PRINT#1,"MSX"
140 CLOSE 1
150 FOR F=1 TO 3000:NEXT F
160 SCREEN 0
170 PRINT#2,"EXPERT"
180 PRINT#2,"MSX"
190 CLOSE 2
```

*MERGE

Carrega um programa armazenado em código ASCII e o une com o programa que está na memória do computador.

- **FORMATO:** MERGE "nome do dispositivo, nome do arquivo"
- **EXEMPLO:** MERGE"CAS:TESTE"
- **FUNÇÃO:** Une um programa que esteja armazenado em código ASCII (em um dispositivo externo) com um programa que esteja na memória do computador.

  Importante: as linhas do programa que se encontram no dispositivo externo serão simplesmente inseridas no programa residente, em seqüência. Se os números das linhas do programa externo coincidirem com os números das linhas do programa residente, as linhas deste serão substituídas pelas linhas do programa externo.

- **PROGRAMA EXEMPLO:**

```
10 'Programa da RAM          5 'Programa da FITA
20 PRINT"RAM .... 20"        15 PRINT"fita _ 15"
30 PRINT"RAM .... 30"        25 PRINT"fita _ 25"
40 PRINT"RAM .... 40"        35 PRINT"fita _ 35"
50 PRINT"RAM .... 50"        45 PRINT"fita _ 45"
60 PRINT"RAM .... 60"        55 PRINT"fita _ 55"
70 PRINT"RAM .... 70"        65 PRINT"fita _ 65"
80 PRINT"RAM .... 80"        75 PRINT"fita _ 75"
                             80 PRINT"fita _ 80 *"

          MERGE"CAS:FITA"

5 'Programa da FITA
10 'Programa da RAM
15 PRINT"fita _ 15"
20 PRINT"RAM .... 20"
25 PRINT"fita _ 25"
30 PRINT"RAM .... 30"
35 PRINT"fita _ 35"
40 PRINT"RAM .... 40"
45 PRINT"fita _ 45"
50 PRINT"RAM .... 50"
55 PRINT"fita _ 55"
60 PRINT"RAM .... 60"
65 PRINT"fita _ 65"
70 PRINT"RAM .... 70"
75 PRINT"fita _ 75"
80 PRINT"fita _ 80 *"
```

MID$

(F) MID$ (middle dollar)

Seleciona parte de *strings* (cadeias alfanuméricas).

- **FORMATO:** MID$(A$,m [,n])
- **EXEMPLO:** A$=MID$("GOEDELESCHERBACH",7,6)
- **FUNÇÃO:** Seleciona um subconjunto de caracteres de A$ (substring), com um comprimento n e começando na posição m. Os valores de m e n devem estar entre 0 e 255. Se n não for inteiro somente sua parte inteira será considerada. Se m for maior que o comprimento da *string* ou se n for zero, a *string* resultante será vazia. Se n for omitido, a *string* resultante será formada pelos caracteres de A$ a partir da posição m.
- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA MID$
20 A$="GOEDELESCHERBACH   ":CLS
30 FOR F=1 TO LEN(A$)-3
40 LOCATE 16,10:FOR D=1 TO 100:NEXT D
50 PRINT MID$(A$,F,3)
60 NEXT F
70 GOTO 30
```

MID$=

Substitui uma parte de uma *string* (cadeia alfanumérica) por elementos de outra *string*.

- **FORMATO:** MID$(A$,m,[,n])=B$
- **EXEMPLO:** MID$(A$,3,4)=B$
- **FUNÇÃO:** Enxerta em A$, a partir do m-ésimo caractere, n caracteres de B$. Os valores de m e n devem estar entre 0 e 255. Se n for omitido, o m-ésimo caractere e os seguintes serão substituídos.
  Importante: o comprimento de A$ não é alterado e a substituição pára quando o último caractere de A$ for substituído.
- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA MID$=
20 A$="GOEDELESCHERBACH GOEDELESCHERBAC
H":CLS
30 C=INT(1+LEN(A$)*RND(1))
40 D=INT(1+LEN(A$)*RND(1))
50 MID$(A$,C,1)=MID$(A$,D,1)
60 LOCATE 3,10:FOR D=1 TO 100:NEXT D
70 PRINT A$
80 GOTO 30
```

MOTOR

MOTOR (motor)

Liga ou desliga o gravador.

- **FORMATO:** MOTOR [ON]
  MOTOR [OFF]

- **EXEMPLO:**

```
MOTOR
MOTOR ON
```

- **FUNÇÃO:** A instrução MOTOR ON liga o gravador e a MOTOR OFF desliga. Se for somente digitado MOTOR o computador desligará o gravador que estiver ligado e ligará se estiver desligado.

- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA MOTOR
20 INPUT "QUANTOS SEGUNDOS O GRAVADOR D
EVE FICAR LIGADO";T
30 MOTOR ON
40 TIME=0
50 IF TIME<T*60 THEN 50
60 MOTOR OFF
```

NEW

Apaga o programa da memória.

- **FORMATO:** NEW

- **EXEMPLO:** NEW

- **FUNÇÃO:** O comando NEW apaga o programa em BASIC e suas variáveis da memória. Ele é usado quando se quer digitar um novo programa. Se existir um programa em linguagem de máquina na memória do computador ele não será apagado com o comando NEW.

- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA NEW
20 PRINT"ESTE PROGRAMA SE AUTO-DESTRUIR
A'EM 10 SEGUNDOS."
30 TIME=0
40 IF TIME=600 THEN NEW
50 GOTO  40
```

OCT$

(F) OTC$ (octonary dollar)

Transforma um dado numérico em uma *string* na forma octal.

- FORMATO: OCT$ (argumento)
- EXEMPLO: A$=OCT$(56)
- FUNÇÃO: A função OCT$ transforma um dado numérico para a forma octal armazenando-o no formato de uma *string.* O valor do agumento deve estar entre -32768 e 65535 e pode ser expresso através de uma constante, uma variável numérica, ou uma variável indexada. Se o argumento for um número negativo, primeiramente calcula-se o valor do argumento subtraido de 65536 e depois é feita a transformação para a forma octal.
- PROGRAMA EXEMPLO:

```
10 REM PROGRAMA OCT$
20 FOR F=0 TO 20
30 PRINT F;".......";OCT$(F)
40 NEXT F
```

ON ERROR GOTO

Desvia o programa para uma determinada linha quando ocorre um erro na execução ou na entrada de dados.

- FORMATO: ON ERROR GOTO nº linha
- EXEMPLO: ON ERROR GOTO 100,200,300,400,500
- FUNÇÃO: A instrução ON ERROR GOTO desvia o programa para uma certa linha definida pelo usuário, em caso de erro na execução do programa ou na entrada de dados. Esta instrução é muito útil para evitar paradas inúteis nos programas.
- PROGRAMA EXEMPLO:

```
10 REM PROGRAMA ON ERROR
20 PRINT
30 ON ERROR GOTO 90
40 PRINT"DIGITE UM NUMERO NEGATIVO."
50 INPUT A
60 PRINT SQR(A)
70 FOR F=1 TO 300: NEXT F
80 RUN
90 PRINT"NAO EXISTE RAIZ REAL."
100 RUN
```

ON - GOSUB

ON-GOSUB (on-goto subroutine)

Desvia o programa para uma sub-rotina condicionada pelo valor de uma variável.

- FORMATO: ON variável GOSUB nº linha [, nº linha, ...]
- EXEMPLO: ON B GOSUB 10,20,30
- FUNÇÃO: A instrução ON-GOSUB funciona analogamente ao ON-GOTO, só que desta vez o programa será desviado para uma sub-rotina. ON-GOSUB também é uma condensação de comandos condicionados e pode ser substituído por:

```
IF B=1 THEN GOSUB 10
IF B=2 THEN GOSUB 20
IF B=3 THEN GOSUB 30
```

- PROGRAMA EXEMPLO:

```
10 REM PROGRAMA ON ... GOSUB
20 INPUT"ENTRE COM UM NUMERO ENTRE 1 E
3";A
30 IF A<1 OR A>3 THEN 20
40 ON A GOSUB 100,200,300
50 RUN
100 PRINT"SUB-ROTINA 1"
110 RETURN
200 PRINT"SUB-ROTINA 2"
210 RETURN
300 PRINT"SUB-ROTINA 3"
310 RETURN
```

ON-GOTO

Desvio condicionado por uma variável.

- FORMATO: ON variável GOTO nº linha [, nº linha, . . .]

- EXEMPLO:

```
ON A GOTO 10,20,30
```

- FUNÇÃO: A instrução ON-GOTO desvia para uma certa linha do programa dependendo do valor de uma variável. O valor da variável que condicionará o desvio deve ser um número inteiro entre 0 e 255. Se o número não for inteiro, as casas decimais serão desprezadas. No exemplo acima ocorre o seguinte: se o valor de A for 1, então o programa desvia para a linha 10, se for 2 para a linha 20, e se 3 para a linha 30. A instrução ON-GOTO pode ser encarada como a condensação dos comandos:

```
IF A=1 THEN GOTO 10
IF A=2 THEN GOTO 20
IF A=3 THEN GOTO 30
```

- PROGRAMA EXEMPLO:

```
10 REM PROGRAMA ON GOTO
20 B=INT(RND(1)*3)+1
30 FOR F=1 TO 400:NEXT F
40 ON B GOTO 60,80,100
50 GOTO 20
60 PRINT"Desvio    1"
70 GOTO 20
80 PRINT"Desvio    2"
90 GOTO 20
100 PRINT"Desvio    3"
110 GOTO 20
```

ON INTERVAL-GOSUB (on interval go to subroutine)

Desvia a execução do programa para uma sub-rotina em um intervalo de tempo periódico.

- **FORMATO:** ON INTERVAL = intervalo GOSUB nº linha

- **EXEMPLO:** `ON INTERVAL=60 GOSUB 100`

- **FUNÇÃO:** A instrução ON INTERVAL GOSUB desvia o programa para uma sub-rotina. No MSX existe um temporizador que causa uma interrupção a cada 1/60 de segundo. O intervalo apontado no formato é exatamente o número de interrupções que serão feitas para que o programa seja desviado. No exemplo anterior acontece o seguinte: o computador espera que sejam feitas 60 interrupções pelo temporizador para executar a sub-rotina da linha 100.

- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA ON INTERVAL
20 ON INTERVAL=50 GOSUB 50
30 INTERVAL ON
40 GOTO 40
50 C=C+1
60 PRINT C
70 RETURN
```

ON KEY GOSUB (on key go to subroutine) **ON KEY GOSUB**

Desvia o programa para uma certa sub-rotina, dependendo da tecla de função que for pressionada.

- **FORMATO:** ON KEY GOSUB nº linha [,nº linha] [,nº linha]...

- **EXEMPLO:** ON KEY GOSUB 10,20,30

- **FUNÇÃO:** A instrução ON KEY GOSUB é utilizada para o desvio do programa para uma sub-rotina quando se pressiona uma das teclas de funções. No exemplo dado acima, caso se pressione a tecla de funções 1, o programa vai para a sub-rotina da linha 10, se for pressionada a segunda tecla de funções o programa vai para a sub-rotina da linha 20, e finalmente, se for pressionada a tecla de funções 3 o programa salta para a sub-rotina da linha 30.

- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA ON KEY
20 ON KEY GOSUB 60,80,100,120,140
30 KEY(5) ON:KEY(4) ON:KEY(3) ON:KEY(2)
ON:KEY(1) ON
40 CLS:PRINT"Pressione uma tecla funcio
nal."
50 GOTO 50
60 PRINT"Tecla 1 pressionada."
70 RETURN
80 PRINT"Tecla 2 pressionada."
90 RETURN
100 PRINT"Tecla 3 pressionada."
110 RETURN
120 PRINT"Tecla 4 pressionada."
130 RETURN
140 PRINT"Tecla 5 pressionada."
150 RETURN
```

ON SPRITE GOSUB (on sprite go to subroutine)

Desvia o programa para uma sub-rotina caso duas figuras móveis na tela se sobreponham.

- FORMATO: ON SPRITE GOSUB [nº de linha]

- EXEMPLO: ON SPRITE GOSUB 500

- FUNÇÃO: A instrução ON SPRITE GOSUB indica em que linha se inicia a sub-rotina caso exista a sobreposição de duas figuras. Esta instrução é muito útil na confecção de jogos de ação.

- PROGRAMA EXEMPLO:

```
10 REM PROGRAMA ON SPRITE
20 SCREEN 2,1
30 SPRITE$(0)=CHR$(&H3C)+CHR$(&H7E)+
  CHR$(&H81)+CHR$(&H81)+CHR$(&HFF)+
 CHR$(&H7E)+CHR$(&H24)+CHR$(&H42)
40 ON SPRITE GOSUB 110
50 SPRITE ON
60 FOR X=0 TO 255
70 PUT SPRITE 0,(X,100),15,0
80 PUT SPRITE 1,(255-X,100),10,0
90 NEXT X
100 RUN
110 SPRITE OFF
120 BEEP
130 SPRITE ON
140 RETURN
```

ON STOP GOSUB (on stop gosub)

ON STOP GOSUB

Instrução que indica para que sub-rotina o programa deve ser desviado caso se pressione CONTROL + STOP.

- **FORMATO:** ON STOP GOSUB número de linha

- **EXEMPLO:** ON STOP GOSUB 100

- **FUNÇÃO:** A instrução ON STOP GOSUB desvia o programa para a sub-rotina indicada pelo número da linha após o GOSUB, quando se pressionam as teclas CONTROL e STOP simultaneamente.

- **PROGRAMA EXEMPLO:**

```
10 REM Programa ON STOP
20 ON STOP GOSUB 70
30 STOP ON
40 SCREEN 0
50 PRINT"Pressione: CONTROL + STOP"
60 GOTO 60
70 PRINT"As teclas CONTROL e STOP foram"
80 PRINT"pressionadas!"
90 PRINT"Para parar o programa, digite"
100 PRINT"a barra de espaços!"
110 IF STRIG(0)=0 THEN 110 ELSE END
120 RETURN
```

ON STRIG GOSUB (on stick trigger go to subroutine)

Desvia para uma sub-rotina caso a barra de espaços ou os botões do *joystick* sejam pressionados.

- **FORMATO:** ON STRIG GOSUB nº linha [,nº linha . . .]

- **EXEMPLO:** ON STRIG GOSUB 100,200,300,400.

- **FUNÇÃO:** A instrução ON STRIG GOSUB tem como função desviar para uma sub-rotina caso a barra de espaços ou algum dos botões dos *joysticks* sejam pressionados. Depois do GOSUB devem vir os números das linhas onde se iniciam as sub-rotinas, na seguinte ordem:

  1. — nº da linha que indica que a barra foi pressionada.
  2. — nº da linha que indica que o botão 1 do joystick A foi pressionado.
  3. — nº da linha que indica que o botão 1 do joystick B foi pressionado.
  4. — nº da linha que indica que o botão 2 do joystick A foi pressionado.
  5. — nº da linha que indica que o botão 2 do joystick B foi pressionado.

- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA ON STRIG
20 ON STRIG GOSUB 90,110,130,150,170
30 FOR F= 0 TO 4
40 STRIG(F) ON
50 NEXT F
60 CLS
70 PRINT"Pressione a barra de espaços o
u um dos botões dos joysticks."
80 GOTO 80
90 PRINT"Barra de espaços pressionada."
100 RETURN
110 PRINT"Botão 1 pressionado (joystick
 A)"
120 RETURN
130 PRINT"Botão 1 pressionado (joystick
 B)"
140 RETURN
150 PRINT"Botão 2 pressionado (joystick
 A)"
160 RETURN
170 PRINT"Botão 2 pressionado (joystick
 B)"
180 RETURN
```

OPEN

Abre um arquivo e especifica modo leitura/escrita.

- **FORMATO:** OPEN "nome do dispositivo [nome do arquivo]"
FOR modo AS [#] número do arquivo

- **EXEMPLO:** OPEN"CAS:MICRO" FOR INPUT AS # 4

- **FUNÇÃO:** Abrir arquivo com um número especificado e realizar operações de entrada/saída para um dispositivo especificado. Os dispositivos são:

Cassete . . . . CAS:

Tela no modo de texto . . . . CRT:

Tela no modo gráfico . . . . GRP:

Impressora . . . . LPT:

O nome do arquivo pode ser formado por até seis caracteres (os excedentes são ignorados). O modo pode ser OUTPUT (para escrita) e INPUT (para leitura). O número de arquivo deve estar entre 1 e o número especificado por MAXFILES.

NOTA: CTR:, GRP: e LPT: estão destinados à escrita. Neste caso, portanto, o modo só poderá ser especificado como OUTPUT.

- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA OPEN
20 MAXFILES=1
30 CLS
40 OPEN "GRP:" FOR OUTPUT AS #1
50 SCREEN 2
55 LINE(0,0)-(255,191),5,BF
56 CIRCLE(120,120),50
60 PRINT #1,"EXPERT"
70 FOR I=1 TO 1000:NEXT
80 CLOSE 1
90 SCREEN 0
100 LIST
```

OUT *

Coloca um byte na porta especificada.

- FORMATO: OUT número da porta, expressão
- EXEMPLO: `OUT 13,240`
- FUNÇÃO: Enviar dados diretamente a uma porta especificada.
- PROGRAMA EXEMPLO:

```
10 SCREEN 0:OUT 153,0:OUT 153,0
20 FOR I=0 TO 959
30 OUT 152,65
40 NEXT I
```

(F) PAD (pad) — PAD

Verifica o estado do *touch pad*.

- **FORMATO:** PAD (n)
- **EXEMPLO:** PAD(2)
- **FUNÇÃO:** Mostrar o estado do *touch pad*. Se n estiver entre 0 e 3, PAD(n) mostrará o estado do *touch pad* ligado ao terminal A. Entre 4 e 7, o valor obtido corresponderá ao terminal B

Valor de n		Significado de PAD(n)
A	B	
0	4	0 tocado;-1 não tocado
1	5	coordenada x do lugar tocado
2	6	coordenada y do lugar tocado
3	7	0 interruptor pressionado -1 interruptor não pressionado

- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA PAD
20 CLS
30 C=0
40 COLOR,C,C
50 IF PAD(3)=0 THEN GOTO 40
60 C=C+1:IF C=15 THEN C=0
70 GOTO 40
```

PAINT

PAINT (paint)

Preenche uma certa área com uma cor especificada pelo usuário.

- FORMATO: PAINT [STEP] (X,Y) [, cor da pintura] [,cor do limite da área]

- EXEMPLO: PAINT(70,70),10,10

- FUNÇÃO: A instrução PAINT colore com a cor especificada pelo usuário uma área pré-definida. O valor da coordenada X pode variar de 0 até 255 enquanto a coordenada Y pode variar de 0 até 191. No modo de alta resolução (SCREEN 2) a cor da linha que limita a área deve ser igual à cor que preencherá a área. Isto não acontece no modo de menor resolução (SCREEN 3). Para maiores detalhes sobre o STEP veja o comando PSET.

- PROGRAMA EXEMPLO:

```
10 REM PROGRAMA PAINT
20 SCREEN 2
30 FOR F=1 TO 90
40 X=RND(1)*256
50 Y=RND(1)*192
60 C=RND(1)*14+2
70 R=RND(1)*30
80 CIRCLE (X,Y),R,C
90 PAINT (X,Y),C,C
100 NEXT F
```

Apresenta o valor determinado pelo *paddle*.

- **FORMATO:** PDL (n)
- **EXEMPLO:** PDL(3)
- **FUNÇÃO:** Transforma em dados numéricos as posições de um *paddle*. Se n for um número ímpar os dados obtidos serão correspondentes ao *paddle* conectado ao terminal A. Se n for par os dados serão provenientes do terminal B. O valor de n pode estar entre 0 e 12 e os valores obtidos estarão entre 0 e 255.
- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA PDL
20 CLS
30 C=0
40 COLOR,C,C
50 IF PDL(1)=0 THEN GOTO 40
60 C=C+1:IF C=15 THEN C=0
70 GOTO 40
```

PEEK *

(F) PEEK (peek)

Apresenta o valor armazenado em um endereço da memória.

- FORMATO: PEEK (endereço)
- EXEMPLO: PEEK(1020)
- FUNÇÃO: Determina o valor do byte armazenado em um determinado endereço da memória. Este endereço pode ser uma constante, variável numérica, variável indexada ou expressões.
- PROGRAMA EXEMPLO:

```
10 REM PROGRAMA PEEK
20 FOR F=0 TO 8191
30 PRINT F;"......"PEEK(F)
40 FOR G=1 TO 200:NEXT G
50 NEXT F
```

PLAY (play)

PLAY

Toca seqüências de notas e/ou acordes musicais compostos de uma a três notas simultaneamente, com tempo, oitava, duração, tom e volume programável.

- **FORMATO:** PLAY subcomandos

- **EXEMPLO:** PLAY"s0m5000v15cdefgab"

- **FUNÇÃO:** Gerar, utilizando o Gerador Programável de Som (PSG), seqüências musicais compostas de até três notas simultâneas, especificadas por subcomandos. Esses subcomandos podem ser representados por *strings* dentro de " " ou por variáveis *string*. Por exemplo:

```
10 PLAY"C"
20 GOTO 10
```

toca uma nota repetidamente

```
10 PLAY"G","E","B"
20 GOTO 10
```

toca um acorde com três notas

```
10 M$="GEB"
20 N$="EGB"
30 O$="BEG"
```

atribui à *string* uma seqüência de notas musicais

```
40 PLAY M$,N$,O$
```

toca as seqüências utilizando simultaneamente os três canais de som

```
50 M$="GEB"
60 PLAY M$,"O2EGB",
"O6BEG"
```

três canais de som em diferentes oitavas

SUBCOMANDOS DA FUNÇÃO PLAY

SUBCOMANDOS	VALORES PERMITIDOS	FUNÇÃO
Tn (tempo)	de 32 a 255	Determina o andamento da música. O valor inicial é T120.
On (oitava)	de 1 a 8	Determina uma das 8 oitavas do MSX. Por exemplo, se for especificado O5, estas serão as notas que poderão ser tocadas: [pauta musical com notas] O valor inicial é O4.

SUBCOMANDOS	VALORES PERMITIDOS	FUNÇÃO
Ln (duração)	de 1 a 64	Determina a duração da nota. L_1 L_2 L_4 L_8 L_{16} L_{32} L_{64} O valor inicial é L4.
Nn (nota)	de 0 a 96	Especifica uma nota musical, por exemplo, N41 é: N0 é uma pausa. Quando n aumenta de uma unidade a nota sobe meio tom.
A-G An-Gn	de 1 a 64	Especifica a nota musical dentro de uma oitava pré-determinada. C# C+ D- / D# D+ E- / F# F+ G- / G# G+ A- / A# A+ B- / C D E F G A B. O sinal "#" (ou "+") colocado ao ao lado da nota especifica um sustenido. O sinal "-" determina um bemol. PLAY"C-" PLAY"C" PLAY"C#" — três notas separadas por meio tom. A duração da nota pode ser fixada através de n. (D4 é o mesmo L4D). Quando omitido, a duração será a determinada por Ln.

SUBCO-MANDOS	VALORES PERMITIDOS	FUNÇÃO
Rn (pausa)	de 1 a 64	Determina uma pausa R_1 R_2 R_4 R_8 R_{16} R_{32} R_{64}
• (ponto)		Aumenta a duração de uma nota ou de uma pausa em 50%. C4. =
Vn (volume)	de 0 a 15	Determina o volume. O volume aumenta com o valor de n. O valor inicial é V8.
Mn (período do envelope)	de 0 a 65535	Determina o período da variação de volume durante a execução da nota. Veja a função SOUND, nota (i) para mais informações.
Sn (forma do envelope)	de 0 a 15	Determina o formato do envelope. Para maiores detalhes veja a função SOUND (tabela de envelopes e nota (j)).

- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA PLAY
20 PLAY"S0M10000V15T180"
30 PLAY"L4O3GFL8EDECO2BABG"
40 PLAY"O3L4CEAGBO4L3CO3"
50 PLAY"L8AGFEL3EL4DGBO4C"
60 PLAY"L6CEL4DL6DFL4ECFEDE"
70 PLAY"L6FEDCL4CO3B"
80 PLAY"O4L4CC#L3DL6DFL4E"
90 PLAY"L6EGL4FDO3GBO4C"
```

POINT

(F) POINT (point)

Obtém o código de cor de um ponto especificado na tela.

- **FORMATO:** POINT(coluna,linha)
- **EXEMPLO:** POINT(143,77)
- **FUNÇÃO:** A função POINT obtém o código de cor de um dado ponto da tela. Os valores da coluna e da linha devem ser uma constante, uma variável ou uma expressão que resulte um número entre -32767 e 32767. Se esse valor estiver fora da faixa correspondente à tela, o valor obtido por POINT será -1.
- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA POINT
20 SCREEN 3
30 FOR I=1 TO 25
40 X=INT(RND(1)*255)
50 Y=INT(RND(1)*191)
60 PSET(X,Y),8
70 NEXT I
80 FOR X=1 TO 250 STEP 4
90 FOR Y=1 TO 191 STEP 4
100 C=POINT(X,Y)
110 IF C=8 THEN PSET(X,Y),14
120 NEXT Y,X
130 FOR B=1 TO 500:NEXT B
```

POKE

Escreve um dado numérico em um certo endereço da memória.

- **FORMATO:** POKE endereço, expressão
- **EXEMPLO:** POKE &HE111,201
- **FUNÇÃO:** A instrução POKE escreve dados numéricos em um certo endereço da memória. Tanto a memória como o dado podem ser escritos em decimal ou hexadecimal. Caso você use números hexadecimais, lembre-se de precedê-los por &H. O valor do endereço pode variar de -32768 (decimal) a 65535 (decimal) e o valor do dado de 0 (decimal) a 255 (decimal).
- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA ESPELHO
20 CLS
30 POKE 40001!,ASC("E")
40 POKE 40002!,ASC("T")
50 POKE 40003!,ASC("N")
60 POKE 40004!,ASC("E")
70 POKE 40005!,ASC("I")
80 POKE 40006!,ASC("D")
90 POKE 40007!,ASC("A")
100 POKE 40008!,ASC("R")
110 POKE 40009!,ASC("G")
120 FOR I=40009! TO 40001! STEP-1!
130 PRINT I, "....."CHR$(PEEK(I))
140 NEXT I
150 END
```

POS

(F) POS (position)

Indica a abcissa X que o cursor ocupa.

- **FORMATO:** POS (X)
- **EXEMPLO:** PRINT POS(0)
- **FUNÇÃO:** A função POS indica a abcissa que o cursor ocupa, sendo que o valor de X entre parênteses não tem significado algum (argumento fictício).
- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA POS
20 CLS
30 INPUT A$
40 PRINT A$;" ";
50 X=POS(5)
60 PRINT"POS=";X
70 GOTO 30
```

PRESET (point reset)

PRESET

Acende ou apaga um ponto nas telas gráficas.

- **FORMATO:** PRESET [STEP] (X,Y) [,cor]
- **EXEMPLO:** PRESET(7,10)
- **FUNÇÃO:** Se for escolhida uma cor diferente da cor do fundo a instrução PRESET funciona exatamente como a PSET. Se nenhuma cor for especificada, porém, será plotado um ponto de cor igual à do fundo, dando a impressão de que o ponto foi apagado.
- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA PRESET
20 SCREEN 3
30 FOR X=0 TO 255
40 PSET (X,25),11
50 PRESET (X,25)
60 NEXT X
70 GOTO 30
```

PRINT

PRINT (print)

Apresenta dados na tela.

- **FORMATO:** PRINT expressão[separador expressão separador]
- **EXEMPLO:** `PRINT "Isto aparecera' na tela"`
- **FUNÇÃO:** O comando PRINT apresenta na tela os dados definidos pelas expressões. Uma expressão pode ser uma constante, uma variável (numéricas ou *strings*). Se a expressão for numérica, basta escrevê-la normalmente, mas se ela for uma *string* deve estar entre aspas. O separador pode ser uma vírgula (,) ou um ponto e vírgula (;). A vírgula faz com que a expressão à sua frente seja apresentada a partir da coluna 0 ou da coluna 14. O ponto e vírgula faz com que a apresentação seja feita logo a seguir o último dado apresentado.
- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA PRINT
20 PRINT "GRADIENTE & ALEPH"
30 GOTO 20
```

PRINT USING

Apresenta dados com um formato específico na tela.

- **FORMATO:** PRINT USING formato;expressão separador . . .
- **EXEMPLO:** PRINT USING"!";"GRADIENTE","ALEPH"
- **FUNÇÃO:** O comando PRINT USING apresenta dados na tela com um formato definido por um dos símbolos da tabela a seguir.

SÍMBOLO	FUNÇÃO	EXEMPLO
!	Apresenta apenas o primeiro carac- de uma *string.*	PRINT USING"!"; A$,B$,C$
"\n-espaço\ "	Apresenta n+2 caracteres de uma *string.*	PRINT USING"\ \" A$;B$;C$
"&"	Apresenta todos os caracteres de uma *string.*	PRINT USING"&"; A$;B$;C$
"#"	Especifica o formato e o número de dígitos apresentados de um dado numérico.	PRINT USING"### # .## ";455.43557
"+"	Acrescenta o sinal + ou – antes ou após dados numéricos conforme eles sejam positivos, nulos ou negativos.	PRINT USING"+ ### ";-233,3434, 675,-13435.232
"-"	Acrescenta o sinal – após números negativos.	PRINT USING"-"; -12312;-13;122
"**"	Preenche com asteriscos os espaços ocupados por um dado numérico.	PRINT USING "** ####. ## ";1.2367
"$$"	Acrescenta o símbolo $ antes de dados numéricos.	PRINT USING"$$ ### ";233,3434,675, 13435.232
"**$"	Acrescenta o símbolo $ antes de dados numéricos e preenche com asteriscos os espaços não ocupados.	PRINT USING"** $ ### ";233,3434, 675,13435,232
","	Acrescenta uma vírgula a cada três dígitos à esquerda do ponto decimal.	PRINT USING "####,. ## "; 1235.122,1133,11378
"^^^^"	Apresenta dados numéricos com ponto flutuante.	PRINT USING"### ## ^^^^";1235,122, 1133,11378

PRINT

PRINT# (print number)

Escreve dados em arquivo aberto por OPEN.

- FORMATO: PRINT# número de arquivo, expressão

- EXEMPLO: PRINT #2,"ACROS"

- FUNÇÃO: Introduzir dados em arquivo aberto por OPEN. O número de arquivos deve estar entre 1 e o especificado por MAXFILES. A expressão pode ser constituída de constantes, variáveis numéricas, variáveis indexadas, alfanuméricas ou suas expressões.

- PROGRAMA EXEMPLO:

```
10 REM PROGRAMA PRINT #
20 MAXFILES=2
30 OPEN "GRP:" FOR OUTPUT AS #1
40 OPEN "CRT:" FOR OUTPUT AS #2
50 SCREEN 2
60 LINE(10,10)-(240,190),9,BF
70 CIRCLE(128,80),60,8
71 PSET(110,70),6
80 PRINT #1,"EXPERT"
90 CIRCLE(128,100),18,8
91 PSET(119,100),3
100 PRINT #1,"MSX"
110 FOR I=1 TO 3000:NEXT
120 CLOSE 1
130 SCREEN 0
140 LIST
```

PRINT#USING

Escreve dados com o formato desejado em um arquivo aberto pelo comando OPEN.

- **FORMATO:** PRINT # número de arquivo USING símbolo de formato, expressão
- **EXEMPLO:** PRINT#2,USING"###.####";A
- **FUNÇÃO:** Esta instrução é usada para introduzir dados em um arquivo, com formato especificado. O número do arquivo deve estar entre 1 e o número de arquivos definido em MAXFILES. A expressão pode ser formada de variáveis numéricas, constantes, variáveis indexadas ou suas expressões.
- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA PRINT # USING
20 MAXFILES=1
30 OPEN "GRP:" FOR OUTPUT AS #1
50 SCREEN 2
60 LINE(60,14)-(196,154),9,BF
70 CIRCLE(128,80),60,12
71 PSET(106,65),9
80 PRINT #1,"EXPERT"
90 CIRCLE(128,100),26,13
91 PSET(110,100),9
100 PRINT #1,USING"##.##";14.11
110 FOR I=1 TO 4000:NEXT
120 CLOSE 1
```

PSET

PSET (point set)

Desenha um ponto na tela de modo gráfico.

- **FORMATO:** PSET [STEP] (coordenada x,coordenada y) [,cor]

- **EXEMPLO:** PSET(129,73),3

- **FUNÇÃO:** Desenha um ponto na tela de modo gráfico, determinado pelas coordenadas x e y. Se as coordenadas forem precedidas pelo STEP (opcional), a origem do sistema de eixos será deslocada do canto superior esquerdo para o último ponto *plotado* antes da execução. A cor deve ser indicada por um inteiro entre 0 e 15. Se for omitida, permanece a cor de fundo atual. As coordenadas x e y podem ser constantes, variáveis numéricas, variáveis indexadas ou suas expressões.

- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA PSET
20 SCREEN 3
30 FOR C=2 TO 8
40 PSET (70,50),C
50 PSET (66,54),C+1
60 PSET (74,54),C+2
70 PSET (62,58),C+3
80 PSET (78,58),C+4
90 PSET (66,62),C+5
100 PSET (74,62),C+6
110 PSET (70,66),C+7
120 FOR F=1 TO 500:NEXT F
130 NEXT C
140 GOTO 30
150 END
```

PUT SPRITE

Torna visível o *sprite* especificado em um endereço determinado do plano de *sprites*.

- **FORMATO:** PUT SPRITE número da camada [[STEP] (coordenada x, coordenada y)], [cor], [número do *sprite*]

- **EXEMPLO:** PUT SPRITE 9,(111,25),4,2

- **FUNÇÃO:** Coloca na tela um *sprite* especificado nas coordenadas desejadas de uma camada também especificada. O número da camada pode variar entre 0 e 31. A coordenada x deve estar entre -32 e 255. A coordenada y deve estar entre -32 e 191. Tanto x quanto y podem ser constantes, variáveis indexadas, variáveis numéricas ou suas expressões. A cor deve ser determinada por inteiros de zero a 15. Se omitida permanece a cor de fundo atual. Se STEP (coordenada x,coordenada y) for omitido fica a posição anteriormente determinada pela última instrução de gráficos. O número do *sprite* depende do número de pontos, ou seja, para 8x8 pontos deve ser um número entre 0 e 255 e para 16x16 pontos, de 0 a 63. Se for omitido será igual ao número da camada a que ele pertence.

- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA PUT SPRITE
20 SCREEN 2,1
30 FOR T=1 TO 8
40 READ A$
50 S$=S$+CHR$(VAL("&B"+A$))
60 NEXT T
70 SPRITE$(1) = S$
80 PUT SPRITE 0,(128,96),8,1: GOTO 80
90 DATA  11000111
100 DATA  11100011
110 DATA  01110011
120 DATA  00111011
130 DATA  11011100
140 DATA  11001110
150 DATA  11000111
160 DATA  11000011
170 GOTO 170
```

READ

READ (read)

Lê os dados que foram armazenados em uma instrução DATA.

- **FORMATO:** READ variável [,variável . . .]
- **EXEMPLO:** READ A,A$,B
- **FUNÇÃO:** A instrução READ lê os dados armazenados na instrução DATA. Os dados podem ser numéricos ou alfanuméricos, desde que o tipo da variável coincida com o tipo de dado. O READ normalmente começa a ler os dados a partir da linha DATA de número mais baixo.
- **PROGRAMA EXEMPLO:**

```
5 REM PROGRAMA READ
10 CLS
20 DIM A$(5), B(5)
30 FOR I=1 TO 5
40 READ A$(I), B(I)
50 NEXT I
60 PRINT "NOME", " NOTA": PRINT
70 FOR K=1 TO 5
80 PRINT A$(K), B(K)
90 NEXT K
100 DATA ALDO, 2.75, ROSANA, 3.17, NANC
Y, 3.65, VANIA, 3.96, ANA, 2.98
```

REM (remark)

REM

Introdução de comentários ou observações na listagem do programa.

- **FORMATO:** REM comentário

- **EXEMPLO:** REM programa exemplo

- **FUNÇÃO:** A instrução REM tem como função facilitar o entendimento de um programa através da leitura de sua listagem. A instrução REM pode ser substituída pelo apóstrofo ('), que tem a mesma função. A instrução REM não interferirá na execução do programa, pois é ignorada pelo computador.

- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA REM
20 REM ESTE PROGRAMA NÃO FAZ
30 REM ABSOLUTAMENTE NADA!
40 '  ESTE SIMBOLO PODE SER USADO NO
50 '  LUGAR DO REM!
```

RENUM

RENUM (renumber)

Renumera as linhas do programa.

- **FORMATO:** RENUM [novo nº da 1ª linha] [, a partir de que linha será renumerado] [, incremento]

- **EXEMPLO:** RENUM

- **FUNÇÃO:** O comando RENUM é muito útil para ser usado no final da confecção de um programa para deixá-lo esteticamente melhor. O RENUM renumera as linhas de 10 em 10 a partir da primeira linha, se for digitado sozinho, ou então renumera como você preferir, utilizando os recursos mostrados no formato. O RENUM também renumera as instruções GOTO e GOSUB.

- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA RENUM
20 PRINT 1
30 PRINT 2
40 PRINT 3
50 PRINT 4
60 PRINT 5
70 RENUM 50,10,20
```

RESTORE

Indica a partir de qual linha a instrução READ deve ler os dados contidos na instrução DATA.

- **FORMATO:** RESTORE [nº de linha]
- **EXEMPLO:** RESTORE 100
- **FUNÇÃO:** Normalmente os dados lidos pela instrução READ começam a partir da linha de número mais baixo que contenha a instrução DATA, porém esta ordem pode ser alterada com a instrução RESTORE que indica a partir de que linha vão ser lidos os dados. A instrução RESTORE também é usada para a releitura dos dados.
- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA RESTORE
20 CLS
30 FOR A=1 TO 2
40 READ A,A$:PRINT A,A$
50 NEXTA
60 RESTORE 100
70 FOR A=1 TO 2
80 READ A,A$:PRINT A,A$
90 NEXT A
100 DATA 123,ABC,456,DEF
```

RESUME

RESUME (resume)

Indica a linha de retorno da rotina principal, após a execução da rotina de zero.

- **FORMATO:** RESUME { [0] / número de linha / NEXT }

- **EXEMPLO:** RESUME 89

- **FUNÇÃO:** A instrução RESUME indica a linha de retorno à rotina principal após a execução de um processo de erros comandado por ON ERROR GOTO. O número da linha deve ser um inteiro entre 0 e 65529 e pode ser expresso por uma constante, uma variável numérica ou uma variável numérica indexada. Se um número de linha for especificado, a execução do programa retornará para a linha indicada. Se for especificado 0, o programa retornará para a linha em que ocorreu o erro e se o NEXT for especificado, retornará para a linha seguinte àquela em que ocorreu o erro.

- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA RESUME
20 ON ERROR GOTO 70
30 CLS
40 INPUT"Digite um numero negativo.";A
50 PRINT SQR(A):FOR B=1 TO 500:NEXT B
60 GOTO 30
70 PRINT"Não existe raiz real de numero
 negativo"
80 FOR B=1 TO 1000:NEXT B
90 RESUME 30
```


```
10 REM PROGRAMA RESUME
20 ON ERROR GOTO 70
30 CLS
40 INPUT"Digite um numero negativo.";A
50 PRINT SQR(A):FOR B=1 TO 500:NEXT B
60 GOTO 30
70 PRINT"Não existe raiz real de numero
 negativo"
80 FOR B=1 TO 1000:NEXT B
90 RESUME 30
```

(F) RIGHT$ (right dollar)

RIGHT$

Separa um pedaço de uma *string*, começando pela direita.

- **FORMATO:** RIGHT$ (*string*, nº de caracteres que serão destacados)
- **EXEMPLO:** PRINT RIGHT$("AGUARDENTE",7)
- **FUNÇÃO:** A função RIGHT$ destaca um certo número de caracteres de uma *string*. No exemplo acima, o resultado será a palavra ARDENTE, pois ela é formada pelos sete últimos caracteres de AGUARDENTE.
- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA RIGHT$
20 CLS
30 INPUT " ESCREVA UMA PALAVRA";W$: PRI
NT
40 PRINT "A ULTIMA LETRA E'...";RIGHT$(
W$,1): PRINT
```

RND

(F) RND (random)

Escolhe um valor aleatório entre 0 e 1.

- FORMATO: RND (X)
- EXEMPLO: PRINT RND(1)
- FUNÇÃO: A função RND tem como finalidade gerar um número pseudo-aleatório entre 0 e 1. O valor de X varia da seguinte maneira: se X é positivo retorna um valor diferente cada vez que é usado; se X é negativo o valor retornado é sempre o mesmo para cada X negativo; se X é igual a 0 retornará o último número aleatório gerado. Para que esta função gere um número realmente aleatório, deve-se usar (-TIME) como argumento.
- PROGRAMA EXEMPLO:

```
10 REM PROGRAMA RND
20 SCREEN 2
30 FOR F=0 TO 100
40 X=256*RND(1):Y=192*RND(2)
50 CIRCLE(X,Y),1
60 NEXT F
70 GOTO 70
```

RUN

Inicia a execução de um programa a partir de uma linha especificada.

- **FORMATO:** RUN [número de linha]
- **EXEMPLO:** RUN 500
- **FUNÇÃO:** O comando RUN zera todas as variáveis e dá início à execução do programa que está na memória. O número de linha pode ser qualquer inteiro entre 0 e 65529 e,se for omitido,o programa começará a ser executado a partir da primeira linha existente. Para parar a execução tecle STOP . Para continuar tecle STOP novamente. Para interromper definitivamente a execução tecle CTRL + STOP . Por exemplo:

  RUN

  inicia a execução a partir da primeira linha do programa e

  RUN 30

  inicia a execução a partir da linha 30 do programa.

- **PROGRAMA EXEMPLO:**

```
10 ' RUN
20 A%=7*RND(-TIME):B%=7*RND(-TIME)
30 CLS:LOCATE6,10
40 PRINT"Digite a barra de espaços."
50 IFSTRIG(0)=0THENLOCATE5,5:GOTO20
60 PRINT:PRINT:PRINT"Dado 1:";A%
70 PRINT"Dado 2:";B%:PRINT:PRINT
80 PRINT"Digite RETURN para novo lançam
ento."
90 LINE INPUT O$
100 RUN
```

SAVE*

SAVE (save)

Armazena, em um dispositivo especificado, um programa em BASIC.

- **FORMATO:** SAVE"[nome do dispositivo] [nome do arquivo]" [,A]
- **EXEMPLO:** SAVE"CAS:BROCHE"
- **FUNÇÃO:** Armazena o programa em BASIC que está na memória do computador no dispositivo especificado.

Os dispositivos são:

Impressora . . . . LPT:
Cassete . . . . CAS:
Tela modo gráfico . . . . GRP:
Tela modo texto . . . . CRT:

Nota: o nome de arquivo não pode exceder seis caracteres (os excedentes não serão considerados). Por exemplo:

```
SAVE"CAS:PROGRA"
```

armazena no gravador o programa em BASIC e

```
SAVE"CRT:"
```

coloca o programa em BASIC na tela de texto.

SCREEN

Determina: tipo de tela, tamanho do *sprite*, o "liga/desliga" do clic das teclas, velocidade de transferência de dados para o cassete e seleciona o tipo de impressora.

- **FORMATO:** SCREEN [modo], [tamanho do *sprite*], [interruptor de clic das teclas], [velocidade em bauds], [tipo de impressora]

- **EXEMPLO:** SCREEN 1,,0,2

- **FUNÇÃO:** Formata a tela, o *sprite*, o clic das teclas, determina a velocidade de transferência de dados para o cassete e especifica o tipo de impressora.

MODO	*0	texto, 40 caracteres x 24 linhas
	1	texto, 32 caracteres x 24 linhas
	2	gráfico alta resolução
	3	gráfico colorido
TAMANHO DO *SPRITE*	*0	8 x 8 sem ampliar
	1	8 x 8 ampliado
	2	16 x 16 sem ampliar
	3	16 x 16 ampliado
INTERRUPTOR DO CLIC DAS TECLAS	0	desligado
	*1 ou qualquer outro valor	ligado
VELOCIDADE DE TRANSFERÊNCIA	*1	1200 Bauds
	2	2400 Bauds
IMPRESSORA	*0	Padrão ABNT
	1 ou qualquer outro valor	outra impressora

* Nota: Os asteriscos indicam os valores iniciais. Se qualquer uma das opções for omitida,permanecem os valores atuais.

No exemplo anterior, a instrução SCREEN determina modo de 32 caracteres por 24 linhas, clic das teclas desligado e velocidade de transferência de dados para o cassete de 2400 bauds.

- **PROGRAMA EXEMPLO:**

```
20 A$="012345678901234567890123456789012
3456789"
30 SCREEN 0:PRINT A$
40 FOR F=1 TO 1000 : NEXT F
50 SCREEN 1:PRINT A$
60 FOR F=1 TO 1000 : NEXT F
70 SCREEN 2:LINE(9,9)-(99,99),3,BF
80 FOR F=1 TO 1000 : NEXT F
90 SCREEN 3:LINE(9,9)-(99,99),3,BF
100 FOR F=1 TO 1000 : NEXT F
```

SGN

(F) SGN (sign)

Verifica o sinal de um número ou expressão.

- FORMATO: SGN (expressão)
- EXEMPLO: PRINT SGN(-10)
- FUNÇÃO: A função SGN verifica o sinal de um número da seguinte maneira: se o número for positivo o valor fornecido por SGN será igual a 1; se o número for negativo, -1 e,se for igual a zero,retornará o próprio valor zero.
- PROGRAMA EXEMPLO:

```
10 REM PROGRAMA SGN
20 A=34
30 B=0
40 C=-345
50 PRINT A,SGN(A)
60 PRINT B,SGN(B)
70 PRINT C,SGN(C)
```

(F) SIN (sinus) SIN

Calcula o valor do seno de um ângulo.

- **FORMATO:** SIN (ângulo)

- **EXEMPLO:**

```
PRINT SIN(1.234)
```

- **FUNÇÃO:** A função SIN calcula o seno de um ângulo fornecido pelo usuário. É importante lembrar que o computador trabalha com o ângulo em radianos. Caso você queira calcular o seno de um ângulo em graus digite da seguinte maneira:

```
PRINT SIN(angulo*3.1415927/180)
```

- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA SIN
20 CLS
30 SCREEN 2
40 FOR F=0 TO 255 STEP .5
50 PSET(F,80-70*SIN(F*6.141592#/255))
60 NEXT F
70 GOTO 70
```

SOUND

SOUND (sound)

Escreve dados diretamente nos registros do PSG (gerador de som programável).

- FORMATO: SOUND número de registro, expressão
- EXEMPLO: `SOUND 7,8`
- FUNÇÃO: Gerar som programando diretamente os registros do PSG. O PSG dispõe de 3 canais que podem gerar tons com a freqüência desejada pelo usuário. É também possível aplicar ruído a todos esses canais, controlar os envelopes (variação de volume durante a geração de um tom) e a duração de cada nota em cada um dos canais. Em outras palavras, podem ser criados acordes triplos e ruído. O PSG possui 16 registros. Veja na tabela as funções de cada um (os registros 14 e 15 não são utilizados para a composição musical). No comando:

SOUND número de registro, expressão

o número de registro deve ser um inteiro entre 0 e 13. As expressões também são inteiras e devem obedecer aos limites estabelecidos na tabela.

Um registro é um "lugar" no PSG onde se armazenam temporariamente dados para o processamento. Variando o dado em um registro, modifica-se o som produzido.

NÚMERO DO REGISTRO	VALORES PERMITIDOS	FUNÇÃO	
0 1	0 a 255 0 a 15	Determinam a freqüência do canal A	(a)
2 3	0 a 255 0 a 15	Determinam a freqüência do canal B	(b)
4 5	0 a 255 0 a 15	Determinam a freqüência do canal C	(c)
6	0 a 31	Determinam a freqüência do ruído	(d)
7	0 a 63	Seleciona um canal para geração de tons e ruídos	(e)
8	0 a 15	Volume do canal A *	(f)
9	0 a 15	Volume do canal B *	(g)
10	0 a 15	Volume do canal C *	(h)
11 12	0 a 255 0 a 255	Freqüência do gerador de envelope	(i)
13	0 a 14	Seleção da forma do envelope	(j)

* NOTA: Só pode haver geração de envelope se o registro de volume do canal for programado com o valor 16.

NOTAS

(a),(b),(c). . . . . .Para determinar os valores dos registros para uma determinada freqüência, use este programa:

```
10 ' Para R0 ate R5
20 INPUT"Qual a frequencia";F
30 A=3575611#/8192/F
40 H=INT(A)
50 L=INT(.5+256*(A-H))
60 PRINT"Alto (impar)=";H,"Baixo (par)=
";L
```

O valor de H deve ser colocado nos registros ímpares (1 para o canal A, 3 para o canal B e 5 para o canal C) e o valor L deve ser colocado nos registros pares (0 para o canal A, 2 para o canal B e 4 para o canal C). Por exemplo, imagine que você quer programar o canal A com uma freqüência de 1000 Hz. Rode o programa e introduza o número 1000.

```
RUN
Qual a frequencia? 1000
Alto (impar)= 0
Baixo (par)= 112
Ok
```

Portanto, devemos fazer:

```
SOUND 0,112
SOUND 1,0
```

(d) . . . . .Para programar a freqüência do ruído use o programa:

```
10 INPUT"Qual a frequencia do ruido";F
20 R6=3575611#/32/F
30 PRINT"O valor a ser atribuido ao reg
istro","6 e':";INT(R6+.5)
```

```
run
Qual a frequencia do ruido? 9600
O valor a ser atribuido ao registro
6 e': 12
```

Portanto fazendo:

```
SOUND 6,12
```

Teremos um ruído de aproximadamente 9600 Hz.

(e) . . . . Apenas com as instruções anteriores o PSG ainda não pode produzir som. É preciso determinar o volume (veremos a seguir) e habilitar a saída do canal que se está programando. Para especificar o canal utilize a tabela:

SOM		
canal A	canal B	canal C
1	2	4

RUÍDO		
canal A	canal B	canal C
8	16	32

Para programar o registro 7 deve-se somar os valores correspondentes às saídas desejadas e subtrair esse total de 255. Por exemplo: para selecionar só o canal A, o valor é 1. Subtraindo temos:

255 - 1 = 254

Portanto, para selecionar o canal A fazemos:

```
SOUND 7,254
```

Se quiséssemos selecionar os canais A e B para produzir som e A, B e C para produzir ruído, teríamos:

255 - (1+2+8+16+32) = 196

Portanto:

```
SOUND 7,196
```

(f),(g),(h). . . . . Há 16 níveis de volume (Ø a 15). Para programar o volume do canal A, fazemos:

```
SOUND 8,15
```

Volume máximo no canal A

Para os outros canais o procedimento é semelhante, bastando mudar o número do registro (9 para o canal B, 1Ø para o canal C). Para utilizar o gerador de envelope (veremos a seguir) os registros escolhidos devem ser carregados com o valor 16.

(i). . . . . .Os registros 11 e 12 determinam a freqüência do envelope (variação de volume durante a produção de um tom). Para obter o valor destes registros utilize este programa:

```
10 ' Para R11 e R12
20 INPUT"Qual a frequencia";F
30 A=3575611#/131072!/F
40 H=INT(A)
50 L=INT(.5+256*(A-H))
60 PRINT"Alto (12)=";H
70 PRINT"Baixo (11)=";L
```

Execute-o, digitando:

```
RUN
```

Na tela, deverão surgir as mensagens:

```
RUN
Qual a frequencia? 16
Alto (12)= 1
Baixo (11)= 180
Ok
```

Portanto, para obter uma freqüência de envelope de 16 Hz no canal A devemos fazer:

```
SOUND 8,16
SOUND 11,180
SOUND 12,1
```

deixa o canal A sob o controle do gerador de envelope

(j). . . . . .Vários envelopes podem ser selecionados para modular a(s) nota(s) emitida(s). O valor atribuído ao registro 13 seleciona um entre os seguintes:

SOUND

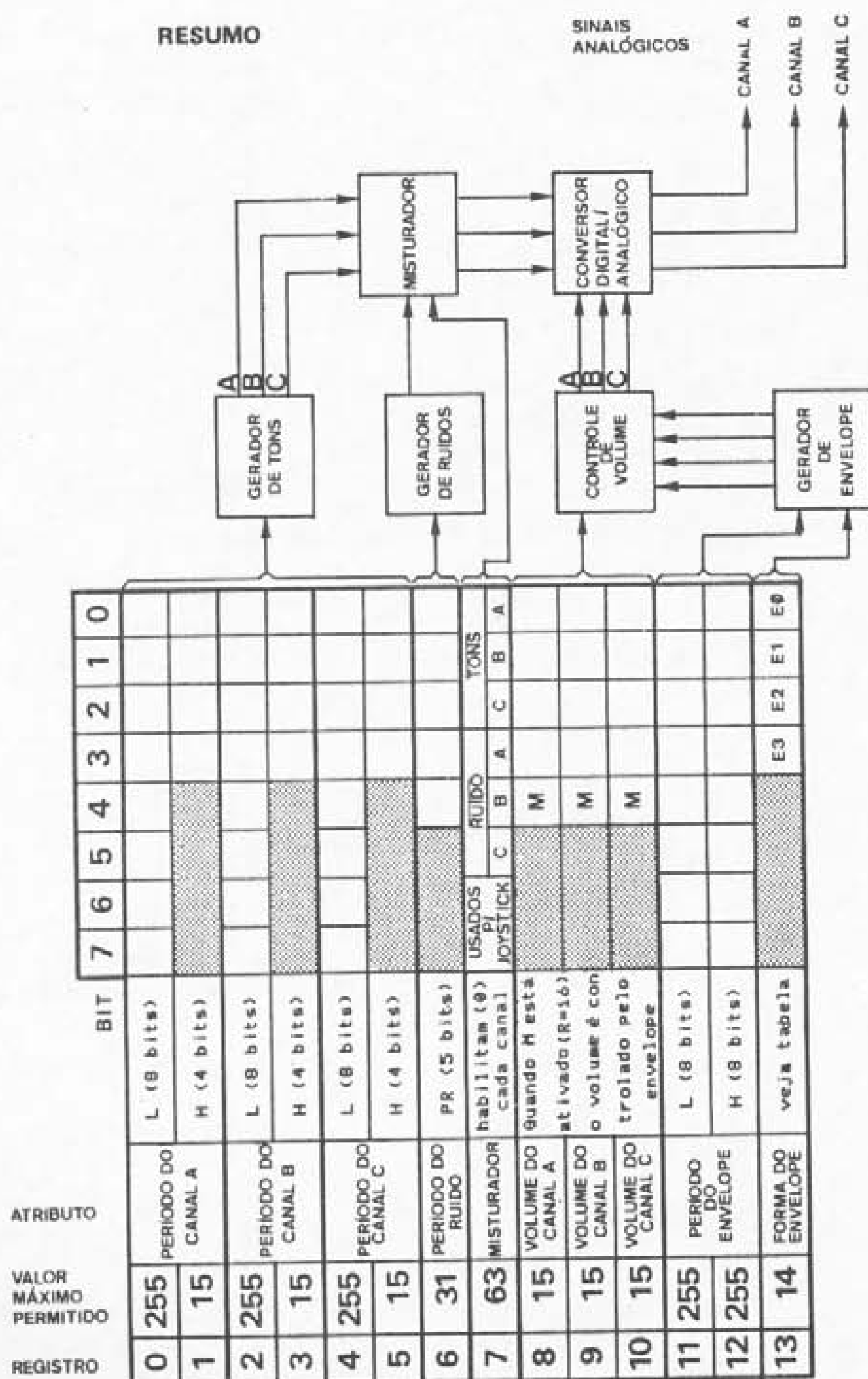
VALOR ATRIBUÍDO AO REGISTRO 13 (valores atribuídos a Sn da função PLAY)	FORMA DO ENVELOPE
Ø,1,2,3 ou 9	
4,5,6,7 ou 15	
8	
10	
11	
12	
13	
14	

Nota: Os valores atribuídos ao registro 13 selecionam os envelopes da mesma forma e com os mesmos valores que o subcomando Sn da função PLAY.

- **PROGRAMA EXEMPLO:**

`100 CLS`	limpa a tela
`110 PRINT "HISTORIA (INACABADA) DA AVIACAO"`	título
`120 SOUND 6,12`	determina a freqüência do ruído (96ØØ Hz)
`130 SOUND 7,246`	habilita saída de som e ruído pelo canal A
`140 SOUND 0,93` `150 SOUND 1,4`	determina a freqüência do canal A (1ØØ Hz)

`160 FOR V = 0 TO 16`	loop para obter valores crescentes de volume. O último valor (16) habilita o gerador de envelope.
`170 FOR T = 0 TO 250:NEXT T`	loop de tempo para fazer o volume subir lentamente
`180 SOUND 8,V`	atribui o volume gerado pelo loop ao canal A
`190 NEXT V`	retorno do loop de volume
`200 SOUND 11,180` `210 SOUND 12,1`	determina a freqüência do envelope (16 Hz)
`220 SOUND 13,14`	seleciona o envelope
`230 FOR T = 0 TO 1000:NEXT T`	loop de tempo. O PSG permanece alguns segundos com o envelope selecionado
`240 SOUND 1,0`	prepara o registro A para produzir tons puros
`250 SOUND 7,254`	habilita a saída de som, sem ruído, pelo canal A
`260 SOUND 8,15`	seleciona o volume máximo
`270 FOR X = 255 TO 0 STEP -1`	loop para variar a freqüência do canal A
`280 SOUND 0,X`	coloca no registro 0 o valor gerado pelo loop (a cada passagem a freqüência aumenta)
`290 NEXT X`	retorno do loop de freqüências
`300 SOUND 0,232` `310 SOUND 1,8`	determina a freqüência do canal A (49 Hz)
`320 SOUND 7,246`	seleciona a saída de ruído e som pelo canal A
`330 FOR T = 0 TO 50:NEXT T`	loop de tempo que mantêm o ruído por alguns décimos de segundo
`340 SOUND 8,0`	seleciona o volume mínimo para o canal A



REGISTRO	VALOR MÁXIMO PERMITIDO	ATRIBUTO	BIT	7	6	5	4	3	2	1	0
0	255	PERÍODO DO CANAL A	L (8 bits)								
1	15		H (4 bits)								
2	255	PERÍODO DO CANAL B	L (8 bits)								
3	15		H (4 bits)								
4	255	PERÍODO DO CANAL C	L (8 bits)								
5	15		H (4 bits)								
6	31	PERÍODO DO RUÍDO	PR (5 bits)								
7	63	MISTURADOR	habilitam (0) cada canal	USADOS P/ JOYSTICK		RUÍDO C	RUÍDO B	RUÍDO A	TONS C	TONS B	TONS A
8	15	VOLUME DO CANAL A	Quando M está				M				
9	15	VOLUME DO CANAL B	ativado (R=16) o volume é con-				M				
10	15	VOLUME DO CANAL C	trolado pelo envelope				M				
11	255	PERÍODO DO ENVELOPE	L (8 bits)								
12	255		H (8 bits)								
13	14	FORMA DO ENVELOPE	veja tabela					E3	E2	E1	E0

(F) SPACE$ (space dollar)

SPACE$

Obtém uma string com o número especificado de espaços.

- FORMATO: SPACE$(argumento)
- EXEMPLO: A$=SPACE$(25)
- FUNÇÃO: Essa função cria uma string com o número de espaços definidos pelo argumento colocado entre parênteses à sua frente. O argumento pode ser uma constante, uma variável simples ou indexada, ou uma expressão matemática, desde que o resultado seja um número entre 0 e 255. Se o resultado for fracionário os dígitos à direita da vírgula (ponto decimal) são ignorados.
- PROGRAMA EXEMPLO:

```
10 REM PROG.  SPACE$
20 FOR F=1 TO 22
30 A$=SPACE$(F)
40 PRINTA$;"ALEPH"
50 NEXT F
```

SPC

(F) SPC (space)

Obtém um certo número de espaços.

- **FORMATO:** SPC(número)
- **EXEMPLO:** SPC(21)
- **FUNÇÃO:** Essa função deve ser usada com as instruções PRINT ou LPRINT e serve para inserir espaços entre expressões a serem apresentadas.
- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA SPC
20 PRINT"GRADIENTE";SPC(25);"ALEPH"
```

(sprite on/off/stop)

SPRITE ON/OFF/STOP

Estas três funções respectivamente, habilitam, desabilitam e adiam um desvio para uma interrupção por sobreposição.

- **FORMATO:**

```
SPRITE ON
SPRITE OFF
SPRITE STOP
```

- **EXEMPLO:**

```
SPRITE ON
SPRITE OFF
SPRITE STOP
```

- **FUNÇÃO:** A instrução SPRITE ON habilita uma interrupção por sobreposição de figuras na tela caso exista uma instrução ON SPRITE GOSUB. A instrução SPRITE OFF desabilita (ou desliga) a função do SPRITE ON. Finalmente a instrução SPRITE STOP adia a interrupção até que seja encontrada uma instrução SPRITE ON.

- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA SPRITE ON/OFF/STOP
20 SCREEN 2,1
30 SPRITE$(0)=CHR$(&H3C)+CHR$(&H7E)+CHR
$(&H81)+CHR$(&H81)+CHR$(&HFF)+CHR$(&H7E
)+CHR$(&H24)+CHR$(&H42)
40 ON SPRITE GOSUB 110
50 SPRITE ON
60 FOR X=0 TO 255
70 PUT SPRITE 0,(X,100),8,0
80 PUT SPRITE 1,(255-X,100),10,0
90 NEXT X
100 GOTO 40
110 SPRITE OFF
120 BEEP
130 SPRITE ON
140 RETURN
```

SPRITE$

SPRITE$ (sprite dollar)

Define os dados de um "sprite."

- **FORMATO:** SPRITE$ (número de *sprite*) = *string*
- **EXEMPLO:** SPRITE$(31)=A$
- **FUNÇÃO:** Definir *sprites*. O número do *sprite* pode estar entre 0 e 255 se ele for de 8x8 pontos e de 0 a 63 se ele for de 16 por 16 pontos. O tamanho do *sprite* e o tamanho de cada pixel devem ser definidos por uma instrução SCREEN.
- **PROGRAMA EXEMPLO:**

```
10 '   SPRITE's
20 FORE=0TO3:D=E+1:KEYOFF
30 SCREEN 1,E:DIM S$(4)
40 LOCATE 4,2:PRINT"SCREEN 1 ,";E:LOCAT
E8,6:PRINT"<-SPRITE 5"
50 FOR G= 1 TO 4
60 FOR F=1 TO 8
70 READ K$
80 S$(G)=S$(G)+CHR$(VAL("&H"+K$))
90 NEXT F
100 NEXT G
110 SPRITE$(1)=S$(1)
120 SPRITE$(2)=S$(2)
130 SPRITE$(3)=S$(3)
140 SPRITE$(4)=S$(4)
150 SPRITE$(5)=S$(1)+S$(2)+S$(3)+S$(4)
160 PUT SPRITE 1,(90,90),3,1
170 PUT SPRITE 2,(90,90+8*D),4,2
180 PUT SPRITE 3,(90+8*D,90),7,3
190 PUT SPRITE 4,(90+8*D,90+8*D),8,4
200 PUT SPRITE 5,(44,44),15,5
205 LOCATE 2,20:PRINT"Digite a barra de
 espacos!"
210 IFSTRIG(0)=0THEN210ELSERESTORE:ERAS
ES$
220 NEXTE:RUN
230 DATA FF,89,99,89,89,89,9D,FF
240 DATA FF,99,A5,85,99,A1,BD,FF
250 DATA FF,BD,85,89,85,A5,99,FF
260 DATA FF,A5,A5,A5,BD,85,85,FF
```

(F) SQR (square root) — SQR

Fornece o valor da raiz quadrada de um número dado pelo usuário.

- **FORMATO:** SQR (expressão)
- **EXEMPLO:** PRINT SQR(16)
- **FUNÇÃO:** A função SQR tem por finalidade dar o valor da raiz quadrada do número entre parênteses. É importante salientar que o número deve ser maior ou igual a zero, caso contrário o computador acusará erro.
- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA SQR
20 FOR A=1 TO 100
30 PRINT A,SQR(A)
40 NEXT A
```

STICK

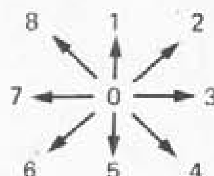
(F) STICK (stick)

Mostra o estado dos *joysticks* e das teclas que movem o cursor.

- **FORMATO:** STICK (N)
  se N=0 – Teclado
  se N=1 – Joystick A
  se N=2 – Joystick B

- **EXEMPLO:** A=STICK(0)

- **FUNÇÃO:** A função STICK tem por finalidade mostrar a direção em que estão se movendo os *joysticks* e as teclas que movem o cursor. A direção é dada pelo seguinte esquema:



- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA STICK
20 ON ERROR GOTO 30
30 SCREEN 3
40 X=0:Y=0
50 PSET(X,Y),6
60 C=STICK(0)
70 IF C=1THENY=Y-1
80 IF C=2THENY=Y-1:X=X+1
90 IF C=3THENX=X+1
100 IF C=4THENY=Y+1:X=X+1
110 IF C=5THENY=Y+1
120 IF C=6THENY=Y+1:X=X-1
130 IF C=7THENX=X-1
140 IF C=8THENY=Y-1:X=X-1
150 GOTO 50
```

STOP

Pára a execução do programa.

- FORMATO: STOP
- EXEMPLO: STOP
- FUNÇÃO: Interrompe a execução do programa na linha onde se encontra. A execução do programa pode ser continuada digitando-se o comando CONT.
- PROGRAMA EXEMPLO:

```
10 REM PROGRAMA STOP
20 WIDTH 40
30 CLS
40 FOR G=1 TO 200
50 CO=RND(1)*40
60 LOCATE CO,23
70 PRINT "*"
80 FOR F=1 TO 20:NEXT F
90 NEXT G
100 LOCATE 10,23
110 PRINT"STOP sera executado"
120 FOR F=1 TO 500:NEXT F
130 STOP
140 RUN
```

STOP ON/OFF/STOP (stop on/off/stop)

Permite, proíbe ou retém a execução de uma sub-rotina definida por ON STOP GOSUB mediante a digitação das teclas CONTROL e STOP.

- FORMATO:

```
STOP ON
STOP OFF
STOP STOP
```

- EXEMPLO:

```
STOP ON
STOP OFF
STOP STOP
```

- FUNÇÃO: Esses comandos permitem, proíbem ou retêm a execução de uma sub-rotina especificada por uma instrução ON STOP GOSUB mediante a digitação simultânea das teclas CONTROL e STOP.

- PROGRAMA EXEMPLO:

```
10 REM PROGRAMA STOP
20 STOP ON
30 ON STOP GOSUB 60
40 CLS:PRINT"Pressione CTRL + STOP."
50 GOTO 50
60 PRINT"CTRL + STOP foi pressionado."
70 FOR B=1 TO 500:NEXT B
80 C=C+1
90 IF C=S THEN END
100 RETURN 40
```

(F) STRIG (stick trigger) STRIG

Obtém o estado de toque da barra de espaços e dos disparadores dos joysticks.

- **FORMATO:** STRIG(número)
- **EXEMPLO:** STRIG(1)
- **FUNÇÃO:** Essa função verifica se a barra de espaços do teclado ou os disparadores dos joysticks estão pressionados. O número que serve como seu argumento pode ser qualquer inteiro entre Ø e 4 e cada um deles corresponde a um caso específico.

  Ø . . . barra de espaços
  1 . . . disparador 1 do joystick 1
  2 . . . disparador 1 do joystick 2
  3 . . . disparador 2 do joystick 1
  4 . . . disparador 2 do joystick 2

  Se nenhum deles está pressionado, o valor obtido por STICK é Ø, caso contrário o valor obtido é -1.

- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA STRIG
20 CLS
30 PRINT"PRESSIONE A BARRA DE ESPAÇOS"
40 C=0
50 COLOR,C,C
60 IF STRIG(0)=0 THEN GOTO 50
70 C=C+1:IF C=15 THEN C=0
80 GOTO 50
```

STRIG (n) ON/OFF/ STOP (strig (n) on/off/stop)

Cada uma dessas três instruções respectivamente habilitam, proibem ou retém uma interrupção mediante o pressionamento da barra de espaços ou de um disparador de um dos joysticks.

- **FORMATO:**

STRIG(n) ON
STRIG(n) OFF
STRIG(n) STOP

- **EXEMPLO:**

```
STRIG(0) ON
STRIG(3) OFF
STRIG(2) STOP
```

- **FUNÇÃO:**

A instrução STRIG(n) ON habilita uma interrupção quando for pressionada a barra de espaços ou um dos disparadores dos joysticks; STRIG(n) OFF desabilita a interrupção e STRIG(n) STOP a retém na mesma situação. A escolha da barra ou de um dos disparadores é feita através do número n colocado entre parênteses logo após STRIG. Esse número pode estar entre 0 e 4, correspondendo em cada caso a um disparador ou a barra de espaços.

0 . . . barra de espaços
1 . . . disparador 1 do joystick 1
2 . . . disparador 1 do joystick 2
3 . . . disparador 2 do joystick 1
4 . . . disparador 2 do joystick 2

- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA STRIG(n) ON/OFF/STOP
20 STRIG(0) ON
30 PRINT"DIGITE A BARRA DE ESPAÇOS"
40 ON STRIG GOSUB 80
50 IF C<>6 THEN 40
60 SCREEN 0:PRINT SPC(17);"FIM":END
70 GOTO 40
80 SCREEN 2
90 COLOR C+3,C+4,C+5
100 CIRCLE (128,85),50
110 COLOR C+6,C+7,C+8
120 C=C+1:IF C=6 THEN COLOR 15,1,1
130 RETURN
```

(F) STR$ (convert to string) — STR$

Transforma dados numéricos em alfanuméricos.

- **FORMATO:** STR$ (expressão numérica)
- **EXEMPLO:** A$=STR$(X)
- **FUNÇÃO:** A função STR$ funciona ao contrário da função VAL, ou seja, parte de uma constante ou uma variável numérica e a transforma em alfanumérica (*string*).
- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA STR$
20 CLS
30 A=45
40 B=60
50 A$=STR$(A)
60 B$=STR$(B)
70 PRINT A+B,A$+B$
```

STRING$

(F) STRING$ (string dollar)

Imprime um caractere definido pelo usuário várias vezes.

- FORMATO: STRING$ (nº de vezes que o caractere será repetido, caractere a ser repetido)

- EXEMPLO:

```
PRINT STRING$(30,67)
```

- FUNÇÃO: A função STRING$ tem como objetivo imprimir várias vezes um certo caractere. O mesmo pode ser impresso até 255 vezes, ou seja, o primeiro número entre parênteses pode variar de 0 até 255, enquanto o segundo define o código do caractere que será impresso. Para saber o código do caractere, consulte a tabela de códigos. Pode-se usar STRING$ também com uma *string* ou uma variável alfanumérica, como por exemplo:

```
PRINT STRING$(10,"A")
PRINT STRING$(19,A$)
```

- PROGRAMA EXEMPLO:

```
10 REM PROGRAMA STRING$
20 A$="###___###"
30 B$="GRADIENTE"
40 C$=STRING$(25,A$)
50 D$="ALEPH"
60 E$=B$+C$+D$
70 PRINT E$
```

SWAP

Troca o valor de duas variáveis.

- **FORMATO:** SWAP variável, variável

- **EXEMPLO:** SWAP A,B

- **FUNÇÃO:** A instrução SWAP troca os valores de duas variáveis que podem ser numéricas ou alfanuméricas. É importante notar que só podem ser trocados os valores de variáveis de mesmo tipo, ou seja, numérica com numérica e alfanumérica com alfanumérica.

- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA SWAP
20 A=12
30 B=24
40 PRINT "A=";A,"B=";B
50 SWAP A,B
60 PRINT "A=";A,"B=";B
```

TAB

(F) TAB (tab)

Move o cursor para a direita um certo número de espaços, definido pelo usuário.

- **FORMATO:** TAB (nº de espaços)
- **EXEMPLO:** PRINT TAB(10);"ALEPH"
- **FUNÇÃO:** A finalidade da função TAB é tabular o texto que está sendo impresso tanto na tela quanto na impressora. O número de espaços, que é definido entre parênteses, não deve exceder 255.
- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA TAB
20 FOR A=1 TO 20
30 PRINT TAB(A);A
40 NEXT A
```

(F) TAN (tangent) TAN

Calcula o valor da tangente de um ângulo.

- **FORMATO:** TAN (ângulo)
- **EXEMPLO:**

```
PRINT TAN(35)
```

- **FUNÇÃO:** A função TAN calcula a tangente de um ângulo fornecido pelo usuário. É importante lembrar que o computador trabalha com o ângulo em radianos. Caso você queira calcular a tangente de um ângulo em graus digite da seguinte maneira:

```
PRINT TAN(angulo*3.1415927/180)
```

- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA TAN
20 CLS
30 SCREEN 2
40 FOR F=0 TO 255 STEP .5
50 PSET(F,80-70*TAN((F+127)*6.1432#/255
))
60 NEXT F
70 GOTO 70
```

TIME

TIME (time)

Determina o valor do temporizador interno.

- **FORMATO:** TIME
  TIME = expressão
- **EXEMPLO:** TIME=0
- **FUNÇÃO:** Enquanto o BASIC estiver acionado, o valor de TIME é incrementado em uma unidade a cada 1/60 de segundo aproximadamente. Ao atingir 65535, TIME voltará a zero e recomeçará a contagem. O valor de TIME pode ser modificado a qualquer momento utilizando-se um comando LET. A expressão pode ser uma constante, variável (indexada ou numérica) ou suas expressões.

  NOTA: O temporizador pára quando o processador realiza algumas operações de entrada/saída. Por exemplo, durante a gravação ou leitura de dados no cassete.

- **PROGRAMA EXEMPLO:**

```
10 ' Relogio
20 INPUT "Horas";H
30 INPUT "Minutos";M
40 INPUT "Segundos";S
50 CLS
60 ON INTERVAL=60 GOSUB 90
70 INTERVAL ON
80 GOTO 80
90 LOCATE 12,10
100 S=S+1
110 IF S=60 THEN M=M+1:S=0
120 IF M=60 THEN H=H+1:M=0
130 IF H=24 THEN H=0
140 PRINT USING"##:##:##";H;M;S
150 RETURN
```

TROFF

Desliga o funcionamento do comando TRON.

- **FORMATO:** TROFF
- **EXEMPLO:** TROFF
- **FUNÇÃO:** Depois de dado o comando TROFF, não mais aparecerão os números das linhas que estão sendo executadas, ou seja, é desativado o comando TRON.
- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA TROFF
20 TRON
30 FOR F=1 TO 40
40 PRINT"................................
....."
50 NEXT F
60 TROFF
70 FOR F=1 TO 40
80 PRINT"................................
....."
90 NEXT F
100 GOTO20
```

TRON

TRON (trace on)

Permite visualizar na tela o número da linha que está sendo executada.

- **FORMATO:** TRON
- **EXEMPLO:** TRON
- **FUNÇÃO:** O comando TRON faz com que o número da linha de programa que está sendo executada apareça na tela entre colchetes. O TRON é muito útil na correção de programas. É importante lembrar que os números das linhas executadas só aparecem no modo de texto. Nas telas gráficas eles não aparecerão. O comando TRON é desativado pelo comando TROFF.
- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA TRON
20 TRON
30 PRINT"LINHA 20";
40 PRINT"LINHA 30";
50 FOR F=1 TO 350:NEXT F
60 GOTO 30
```

USR (user) (F)

*USR

Fornece o resultado obtido pela execução de uma rotina em linguagem de máquina, que começa no endereço definido em DEFUSR.

- **FORMATO:** USR [número da rotina]
  (valor a ser transferido)

- **EXEMPLO:** M=USR1(34)

- **FUNÇÃO:** A função USR fornece o resultado obtido após a execução de uma rotina em linguagem de máquina. Os endereços de início das rotinas são definidos por DEFUSR. O número da rotina deve ser um inteiro entre 0 e 9 e, quando omitido, assumirá o valor 0. O valor a ser transferido para a sub-rotina, pode ser expressado por uma constante, uma variável numérica ou então, por uma variável numérica indexada. Veja o item SUB-ROTINAS EM LINGUAGEM DE MÁQUINA na parte 3 deste livro.

- **PROGRAMA EXEMPLO:**

```
10 DATA 3E,41,21,00,00,CD,4D,00,23,E5,D
5,11,BF,03,ED,52,D1,E1,38,F1,C9
20 FOR I=0 TO 20
30 READ A$:POKE 60000!+I,VAL("&H"+A$)
40 NEXT I
50 SCREEN 0:CLS:PRINT "APERTE <RETURN>
PARA EXECUTAR"
60 INPUT B$
70 DEF USR0=60000!
80 X=USR0 (0)
90 FOR T=1 TO 1000:NEXT T:CLS:END
```

VAL

(F) VAL (value)

Transforma dados alfanuméricos em numéricos.

- **FORMATO:** VAL (*string* ou variável alfanumérica)
- **EXEMPLO:** PRINT VAL("10")
- **FUNÇÃO:** A função VAL tem por finalidade transformar dados alfanuméricos em numéricos, sendo que os dados alfanuméricos podem ser uma *string* ou mesmo uma variável alfanumérica. Essa função ignora caracteres que não sejam de algarismos.
- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA VAL
20 A$="10"
30 B$="A55"
40 C$="55A"
50 PRINT A$,VAL(A$)
60 PRINT B$,VAL(B$)
70 PRINT C$,VAL(C$)
```

(F) VARPTR (variable pointer)

VARPTR

Obtém o endereço do byte a partir do qual está armazenado o conteúdo de uma variável.

- **FORMATO:** VARPTR(variável)
- **EXEMPLO:** VARPTR(A$)
- **FUNÇÃO:** Essa função obtém o endereço do primeiro byte a partir do qual está armazenado o conteúdo de uma variável, que pode ser numérica, *string* ou indexada. O valor obtido pode estar entre -32768 e 32767. Se o valor é negativo deve-se acrescentar 65536 a ele para obter o endereço.
- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA VARPTR
20 CLS
30 PRINT"ESTE PROGRAMA MOSTRA COMO A VA
RIAVEL YXESTA ARNAZENADA NA MEMORIA DO
EXPERT."
40 YX=124
50 C=VARPTR(YX)
60 FOR F=C-3 TO C+7
70 PRINT F;"...";PEEK(F);"...";CHR$(PEE
K(F));"..."
80 NEXT F
```

VDP*

VDP (video display processor)

Lê e insere dados diretamente nos registros do VDP.

- FORMATO: VDP(registro)
  VDP(registro)=expressão

- EXEMPLO: VDP(7)=123

- FUNÇÃO: Esse comando lê ou escreve dados nos registros do VDP, responsável pelo gerenciamento da tela do MSX. O registro é especificado por um número inteiro entre 0 e 8 e a expressão pode ser uma constante, uma variável simples ou indexada que resulte um número inteiro entre 0 e 255.

- PROGRAMA EXEMPLO:

```
10 SCREEN0:LOCATE7,10:PRINT"Pressione a
 barra de espaco"
20 FOR I=0 TO 255
30 IF STRIG(0)=0 THEN 30
40 VDP(7)=I
50 NEXT I
60 RUN
```

VPEEK (video ram peek)

*VPEEK

Lê o byte armazenado no endereço especificado na RAM de vídeo.

- **FORMATO:** VPEEK (endereço)
- **EXEMPLO:** PRINT VPEEK(1234)
- **FUNÇÃO:** Obter o valor do byte armazenado em determinado endereço da RAM de vídeo. Esse endereço deve ser indicado por inteiros entre 0 e 16383. O endereço de cada tabela pode ser encontrado com a função BASE.
- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA VPEEK
20 CLS
30 PRINT"ABBCCCDDDDEEEEEFFFFFFGGGGGGGHH
HHHHHHIIIIIIIIIJJJJJJJJJJ"
40 FOR F=1 TO 55
50 PRINT VPEEK(F);"...";
60 NEXT F
```

VPOKE*

VPOKE (video RAM poke)

Insere dados num dado byte da VRAM (RAM de vídeo).

- **FORMATO:** VPOKE endereço,dado
- **EXEMPLO:** VPOKE 929,255
- **FUNÇÃO:** Esse comando insere um dado num byte da VRAM, controlada pelo VDP. O endereço pode estar entre 0 e 16383 e o dado pode estar entre 0 e 255.
- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA VPOKE
20 CLS
30 FOR F=0 TO 255
40 VPOKE F,F
50 NEXT F
60 FOR F=1 TO 1000:NEXT F
70 CLS
80 LIST
```

* WAIT

Espera até que um certo valor surja na porta de entrada/saída.

- **FORMATO:** WAIT número da porta de E/S, expressão1, [expressão 2]

- **EXEMPLO:** `WAIT 21,127`

- **FUNÇÃO:** Um XOR (ou exclusivo) é executado entre os dados da porta de E/S especificada e o valor da expressão 2. Um AND (e) é executado entre o resultado desta operação e o valor da expressão 1. Se o resultado final for Ø,os dados procedentes da porta de entrada/saída serão introduzidos continuamente. As expressões podem assumir qualquer valor entre Ø e 255. Se a expressão 2 for omitida, assumirá o valor Ø.

WIDTH

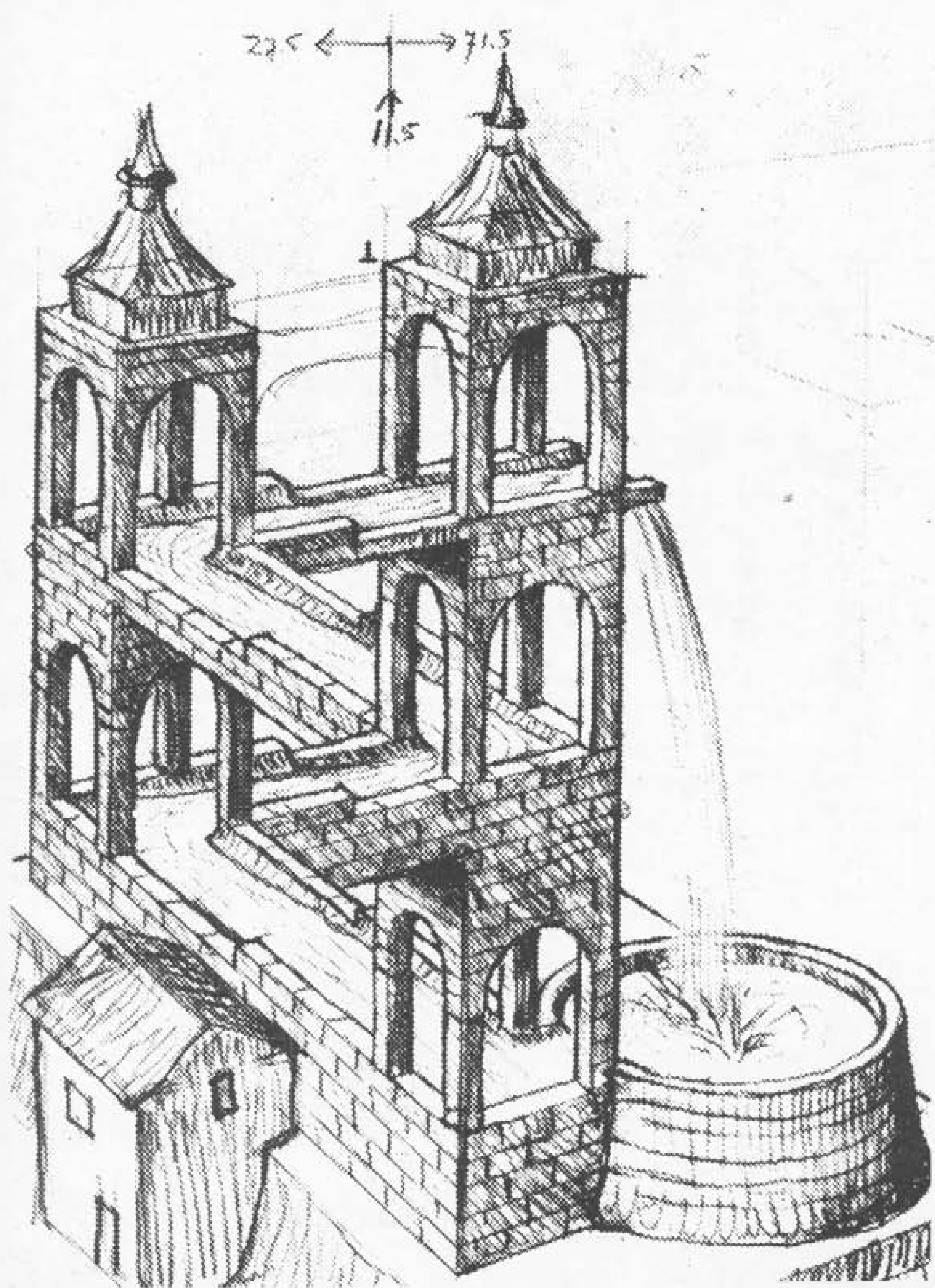
WIDTH (width)

Determina o número de caracteres por linha no modo texto (SCREEN 0 ou 1).

- **FORMATO:** WIDTH (número de caracteres)
- **EXEMPLO:**
- **FUNÇÃO:** Pode-se determinar o número de caracteres por linha na tela do modo texto. No modo texto SCREEN 0, o número de caracteres deve ser um número inteiro entre 0 e 40. No modo texto SCREEN 1, o número de caracteres deve estar entre 1 e 32. Ao ser ligado, o computador coloca 40 colunas no SCREEN 0 (29 no SCREEN 1).
- **PROGRAMA EXEMPLO:**

```
10 REM PROGRAMA WIDTH
20 INPUT"ESTE PROGRAMA SERA LISTADO ASS
IM QUE   FOR EXECUTADO. QUANTAS COLUNAS
 A TELA  DEVERA TER";N
30 WIDTH(N)
40 LIST
```

PARTE III
APLICAÇÕES ESPECIAIS



INTERRUPÇÕES

A interrupção pára a execução de um programa para executar uma sub-rotina chamada normalmente de rotina de interrupção.

Quando desenvolvemos um jogo, por exemplo, é normal usarmos o comando ON STRIG GOSUB para saber se os botões do joystick foram pressionados.

OS COMANDOS DE INTERRUPÇÃO NO MSX

O MSX dispõe de vários comandos de interrupção. A tabela a seguir mostra um resumo destas instruções:

interrupção quando se pressiona:	Instrução de interrupção
uma tecla de função	ON KEY GOSUB
barra de espaços ou botões dos joysticks	ON STRIG GOSUB
CTRL + STOP	ON STOP GOSUB

Além dessas instruções, que interrompem a execução, caso estas teclas sejam pressionadas, existem mais dois comandos de interrupção: ON SPRITE GOSUB, que interrompe o programa quando duas figuras se sobrepõem, e ON INTERVAL GOSUB, que interrompe o programa em espaços periódicos de tempo.

TORNANDO AS INTERRUPÇÕES HABILITADAS

Para que a interrupção realmente aconteça é necessário que seja dado um comando que habiiite (ligue) esta interrupção. Estes comandos são:

KEY (X) ON
STRIG (X) ON
STOP ON
SPRITE ON
INTERVAL ON

EXEMPLIFICANDO ALGUMAS INTERRUPÇÕES

Digite o programa a seguir:

```
10 ON KEY GOSUB 100
20 KEY(1)ON
30 SCREEN 2
40 LINE(50,50)-(200,150),,B
50 GOTO 40
100 REM Sub-rotina
110 BEEP:CLS
120 FORI=1 TO 90 STEP 10
130 CIRCLE(120,100),I
140 NEXT I
150 CLS
160 RETURN
```

Este programa depois de rodado desenhará na tela um retângulo. Caso seja pressionada a tecla F1 o programa se desviará para a linha 100 devido à sentença das instruções de interrupções nas linhas 10 e 20. Na sub-rotina que começa na linha 100 ele desenhará nove círculos concêntricos, e depois retornará para a linha 40, onde novamente o retângulo será desenhado.

Agora pressione duas vezes seguidas a tecla F1 e verifique o que aconteceu.

Você notou que quando foi pressionada a tecla F1 pela segunda vez, a sub-rotina de interrupção não parou, e a nova interrupção só foi feita quando a sub-rotina acabou de ser executada.

Vejamos duas variações deste mesmo programa. Acrescente a linha:

```
105 KEY(1) OFF
```

Execute novamente o programa e pressione a tecla F1. Agora pressione-a de novo. Você percebeu que a interrupção só foi executada uma vez. Isto acontece porque quando é causada a primeira interrupção o programa executa a linha 105 que desabilitará as próximas execuções.

Como uma segunda variação acrescente ao programa a linha:

```
105 KEY(1) ON
```

Com esta modificação, toda vez que for pressionada a tecla F1, será executada a sub-rotina de interrupção. Mesmo que ela já esteja sendo executada, será desviada para seu início. Isto ocorre porque a linha 105, neste caso, reabilita uma nova interrupção.

PROCESSO DE ARQUIVO

Um arquivo é uma região da memória onde podem ser armazenados dados, que possam ser constantemente atualizados.

Um bom exemplo seria um DIÁRIO pessoal. Ele é manuseado constantemente e, além de guardar dados do dia, pode ser consultado também sobre registros anteriores.

No MSX existem comandos que fazem exatamente isto. Antes, porém, vejamos o que é necessário para montar este sistema.

Para se escrever no arquivo, serão necessários os chamados dispositivos de saída (vídeo, impressora, gravador) e para consultar um arquivo precisamos dos dispositivos de entrada (gravador). O gravador funciona tanto como dispositivo de entrada, como saída, pois nele podemos ler e escrever dados. Todas estas entradas e saídas são controladas pelo microprocessador que se encontra dentro das UCP.

Cada um desses dispositivos tem uma abreviatura que será usada quando forem dados os comandos de leitura e gravação de dados. Estas abreviaturas são:

Nome do dispositivo	Abreviatura
gravador cassete	CAS:
tela modo texto	CRT:
tela modo gráfico	GRP:
impressora	LPT:

Os nomes dos arquivos podem conter, no máximo, 6 caracteres (os excedentes serão desprezados).

Os comandos para abrir, ler, gravar e fechar um arquivo são:

OPEN	abre um arquivo
PRINT#	escreve no arquivo
PRINT# USING	escreve no arquivo
INPUT#	lê um arquivo
LINE INPUT#	lê um arquivo
CLOSE	fecha arquivo

GRAVAÇÃO DE DADOS DE UM ARQUIVO

Para gravar dados de um arquivo, deve-se proceder da seguinte maneira:

1) Abre-se o arquivo com a instrução OPEN.
2) Escrevem-se os dados com a instrução PRINT#.
3) Fecha-se o arquivo com a instrução CLOSE.

O formato da instrução OPEN para a gravação de dados é:

OPEN "abreviatura do dispositivo nome do arquivo" FOR OUTPUT AS [#] número do arquivo

Quando se executa isto, estabelece-se que um arquivo será gravado no dispositivo, indicado.

Quando o MSX está lendo ou escrevendo um arquivo, antes de armazená-los, passa os dados para uma memória intermediária (*buffer*). O MSX contém 16 *buffers.* O número do arquivo determina em qual *buffer* serão armazenados provisoriamente os dados.

Depois de aberto o arquivo, com a instrução OPEN, devem se escrever os dados com a instrução PRINT# que tem o seguinte formato:

PRINT# número do arquivo, expressão [, expressão ...]

O número do arquivo deve ser o mesmo que consta da instrução OPEN.

Depois de gravados os dados, será automaticamente gravado um código de "carriage return" (&HOD) e um "line feed" (&H0A), para que os dados escritos seja separados.

Um exemplo de como usar a instrução PRINT# pode ser o seguinte:

```
PRINT#1,A$;",";B$
```

A vírgula entre aspas indica que a variável A$ deve ser separada da B$.

Finalmente, para fechar o arquivo você deve digitar a instrução CLOSE da seguinte maneira:

CLOSE [#] número do arquivo

LEITURA DE DADOS DE UM ARQUIVO

Para se ler um arquivo deve-se proceder da seguinte maneira:

1) Abre-se o arquivo com a instrução OPEN.
2) Lê-se o arquivo com a instrução INPUT#.
3) Fecha-se o arquivo com a instrução CLOSE.

O formato da instrução OPEN para leitura de dados é o seguinte:

OPEN "abreviatura do dispositivo nome do arquivo" FOR INPUT AS [#] número do arquivo

Depois de aberto o arquivo com a instrução OPEN, os dados devem ser lidos com a instrução INPUT#, da seguinte maneira:

INPUT# número do arquivo, variável

Finalmente, deve-se fechar o arquivo com a instrução CLOSE.

SUB-ROTINAS EM LINGUAGEM DE MÁQUINA

O microprocessador utilizado pelos micros da linha MSX é o Z-80A, para o qual podemos escrever programas em linguagem de máquina. Através de instruções do BASIC pode-se também transferir valores para esse programa e iniciar sua execução. Se for necessário é possível voltar ao BASIC, através da instrução RET (return) do Z-80.

PREPARANDO UM PROGRAMA EM LINGUAGEM DE MÁQUINA

1. Limpe uma área da memória para colocar seu programa em LM usando o comando:

   CLEAR n, endereço

   Por exemplo:

```
10 CLEAR 200,&HDFFF
```

   No exemplo foram reservados 200 bytes de memória .

2. Coloque o programa na memória através do comando:

   POKE (endereço), byte

   Por exemplo:

```
20 POKE &HDFFF,&HCD
30 POKE &HE000,&HC3
40 POKE &HE001,&H0
50 POKE &HE002,&HC9
```

   No exemplo, o programa é constituído pelas instruções CALL (&HCD), pelo endereço &H00C3 (&HC3.&H00 – o byte menos significativo vem antes) e por um RETurn (&HC9).

3. Defina o ponto de entrada (onde o Z-80 deve iniciar a execução do programa), utilizando o comando:

   DEFUSR n = endereço

   Por exemplo:

```
DEFUSR 1 = &HDFFF
```

   O BASIC-MSX permite definir 10 pontos de entrada diferentes, o que significa que podemos chamar 10 sub-rotinas diferentes em linguagem de máquina a partir do programa em BASIC. No exemplo definiu-se como início da sub-rotina 1, o endereço &HDFFF.

4. Execute a sub-rotina através do comando:

X = USRn (I)

Por exemplo:

```
40 X = USR1(I)
```

Este programa em BASIC coloca na memória do computador um programa em LM que chama uma sub-rotina da ROM que limpa a tela. Esta sub-rotina está no endereço &H00C3.

PASSANDO VALORES PARA A SUB-ROTINA EM LM

Quando for preciso enviar um valor a uma sub-rotina em LM utiliza-se a forma:

Variável=USR n(I)

O valor enviado será I e se houver valor para retornar ao BASIC este será atribuído à variável. Se fizermos:

```
X = USR1(I)
```

Os dados que indicam o tipo e o valor da variável utilizada são armazenados na memória de acordo com a tabela a seguir, e o endereço do começo da área de armazenamento é indicada pelo registro HL.

O dado no registro A também é armazenado no endereço &HF663.

TIPO DE **I**	REGISTRO **A**	REGISTROS **HL**	ENDEREÇO ONDE É ARMAZENADO O VALOR DE **I**
dupla precisão	8	&HF7F6	&HF7F6–&HF7FD
precisão simples	4		&HF7F6–&HF7F9
inteiro	2		& HF7F8 - & HF7F9

Se a variável for uma *string*:

REGISTRO **A**	REGISTROS **DE**	PONTEIRO DA *STRING*
3	Endereço do ponteiro da *string*	1 byte: Comprimento da *string* 2 e 3 bytes: endereço onde a variável está armazenada.

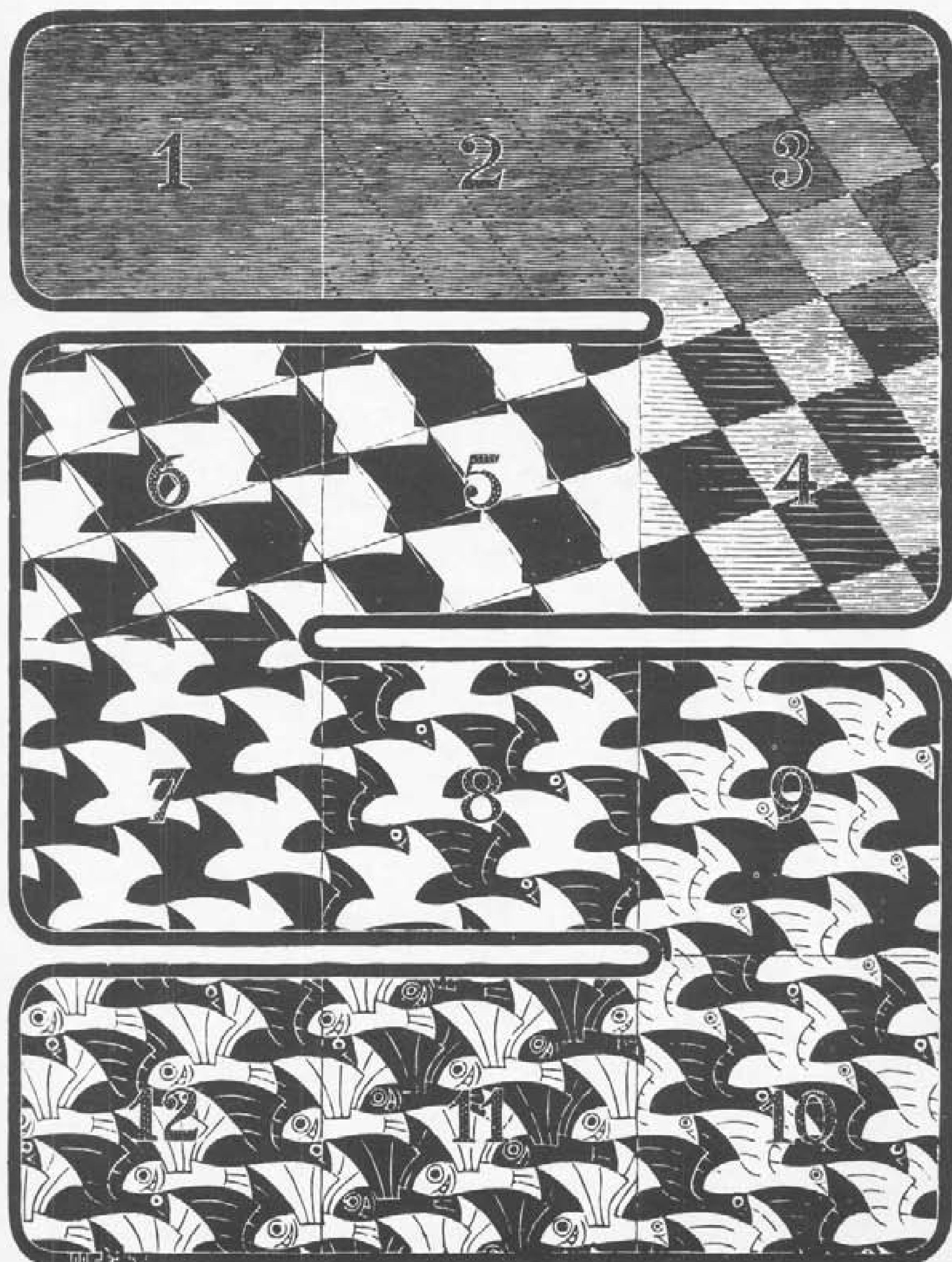
Terminada a sub-rotina, o valor obtido será atribuído à variável X e à memória, e os registros ficarão organizados assim:

TIPO DE RESULTADO	ENDEREÇO &HF633	REGISTROS **DE**	REGISTROS **HL**	ENDEREÇO ONDE O RESULTADO É ARMAZENADO
PRECISÃO DUPLA	8	–	&HF7F6	&HF7F6 - &HF7FD
PRECISÃO SIMPLES	4	–	&HF7F6	&HF7F6 - &HF7F9
INTEIRO	2	–	&HF7F6	&HF7F8 - &HF7F9
STRING	3	Endereço do 1 byte do ponteiro da *string*		Endereço indicado pelos 2 e 3 bytes do ponteiro de *string*

EXEMPLO:

```
5 REM ORDENADOR EM LINGUAGEM DE MAQUINA
10 '               D A D O S
15 DATA 035,035,094,035,086,213,221,225
20 DATA 221,094,254,221,086,255,027,221
25 DATA 229.221,070.000,221,078,003,221
30 DATA 110,004,221,102,005,229,253,225
35 DATA 126,221,110,001,221,102,002,150
40 DATA 056,016,032,058,005,040,055,013
45 DATA 040,008,035,253,035,253,126,000
50 DATA 024,237,221,043,221,043,221,043
55 DATA 221,126,006,221,070,003,221,112
60 DATA 006,221,119,003,221,126,007,221
65 DATA 070,004,221,112,007,221,119,004
70 DATA 221,126,008,221,070,005,221,112
75 DATA 008,221,007,005,024,171,221,225
80 DATA 221,035,221,035,221,035,027,122
85 DATA 179,032,156,201
90 '        INICIO DO PROCESSAMENTO
95 CLEAR 6000,&HF000:DEFINT A-Z
100 DEFUSR=&HF000
105 INPUT"Quantos nomes serão inseridos
";N:DIM N$(N)
110 '   INSERE OS CODIGOS DAS LINHAS
         DATAS NOS BYTES ESPECIFICADOS
115 FOR F=0 TO &H73
120 READ D:POKE &HF000+F,D
125 NEXT F
130 '   RECEBE OS NOMES A SEREM
       ORDENADOS ATRAVES DO TECLADO
135 FOR F=1 TO N: F$=STR$(F)
140 PRINT "Qual o ";F$;CHR$(&HF8);" nom
e";:INPUT N$(F):B$=N$(F)
145 FOR G=1 TO LEN(B$)
150 C$=MID$(B$.G.1):C=ASC(C$)
155 IF C<97 THEN C=C+32:C$=CHR$(C)
160 MID$(B$,G,1)=C$
165 NEXT G
170 MID$(B$,1,1)=CHR$(ASC(LEFT$(B$.1))-
32):N$(F)=B$
175 NEXT F
180 '   INICIA A ORDENAÇÃO
185 N$(0)=" "+"":E=VARPTR(N$(0))
190 PRINT "INICIO !":TIME=0
195 E=USR(E):T=TIME
200 FOR F=1 TO N
205 PRINT F;N$(F)
210 NEXT F
215 PRINT:PRINT:PRINT"Tempo de processa
mento:"
220 PRINT INT(1000000#*T/60);"micro-seg
undos"
225 END
```

PARTE IV
APÊNDICES



APÊNDICE A

CARACTERES ASCII

A tabela a seguir fornece o código hexadecimal dos 256 caracteres armazenados na memória ROM do seu EXPERT. Para obter um determinado código você deve ler primeiro a linha e depois a coluna. Por exemplo, o caractere L corresponde ao código ASCII 4C (em hexadecimal).

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		☺	☻	♥	♦	♣	♠	•	◘	○	◙	♂	♀	♪	♫	☼
1	┼	┴	┬	┤	├	┼	│	─	┌	┐	└	┘	╳	╱	╲	┼
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	¦	}	~	▴
8	Ç	ü	é	â	Á	à	¨	ç	ê	í	ó	ú	Â	Ê	Ô	À
9	É	æ	Æ	ô	ö	ò	û	ù	ÿ	Ö	Ü	¢	£	¥	₧	ƒ
A	á	í	ó	ú	Ã	Ñ	ª	º	¿	⌐	¬	½	¼	¡	«	»
B	Ã	ã	Ĩ	ĩ	Õ	õ	Ũ	ũ	Ĳ	ĳ	¾	∽	◇	‰	¶	§
C	▁	▚	▆	▔	▬	▀	▎	▞	▊	▕	▉	▨	▧	▼	▲	▶
D	◀	⧗	⧓	▘	▗	▝	▖	▩	Δ	ǂ	ω	█	▄	▌	▐	▀
E	α	β	Γ	π	Σ	σ	µ	γ	Φ	θ	Ω	δ	∞	ø	∈	∩
F	≡	±	≥	≤	⌠	⌡	÷	≈	°	∙	·	√	ⁿ	²	▮	

APÊNDICE B
IMPRESSORA

Exitem duas tabelas de caracteres que podem ser selecionadas ao se usar uma impressora: a da ABNT (que já está ativada ao ligarmos o Expert) e a do MSX. Para selecioná-las, basta alterar o conteúdo da posição &HF417 da memória ou o último atributo do comando SCREEN, conforme o quadro a seguir:

Padrão ABNT	Padrão MSX
POKE &HF417,0	POKE &HF417,1
SCREEN ,,,,0	SCREEN ,,,,1

Na tabela de conversão MSX-ABNT a seguir estão relacionados os códigos (em hexadecimal) dos caracteres listados. Para qualquer caractere de código maior que 127 (&H7F) que não conste desta tabela, será enviado para a impressora um espaço em branco, de código 32 (&H20).

TABELA DE CONVERSÃO MSX – ABNT

MSX	ABNT		MSX	ABNT		MSX	ABNT		MSX	ABNT	
AD	A1	¡	AC	BC	¼	89	CD	Í	B1	E3	ã
BF	A7	§	AB	BD	½	A5	D1	Ñ	87	E7	ç
A6	AA	ª	BA	BE	¾	8A	D3	Ó	82	E9	é
AE	AB	«	A8	BF	¿	8E	D4	Ô	88	EA	ê
F8	B0	°	8F	C0	À	B4	D5	Õ	A1	ED	í
F1	B1	±	84	C1	Á	8B	DA	Ú	A4	F1	ñ
FD	B2	²	8C	C2	Â	9A	DC	Ü	A2	F3	ó
E6	B5	µ	B0	C3	Ã	E1	DF	ß	93	F4	ô
BE	B6	¶	80	C7	Ç	85	E0	à	B5	F5	õ
A7	BA	º	90	C9	É	A0	E1	á	A3	FA	ú
AF	BB	»	8D	CA	Ê	83	E2	â	81	FC	ü

APÊNDICE C

CÓDIGOS DE ERROS

```
Bad file name
```

Nome Incorreto para Arquivo

Uma forma ilegal foi usada para o nome do arquivo com: LOAD, SAVE, KILL, NAME, etc.

```
Bad file number
```

Problemas com Número de Arquivo

Há um termo, ou comando, referente a um número de arquivo, o qual não está aberto (OPEN). Ou então, ele está fora do alcance dos números de arquivo especificados pelo termo MAX-FILE.

```
Can't continue
```

Impossibilidade de Continuar

Foi feita uma tentativa de continuar um programa, o qual:

1. Parou devido a um erro;
2. Tenta ser modificado durante uma parada na execução;
3. Ou, simplesmente não existe continuação.

```
Device I/O error
```

Erro no Dispositivo de I/O

Ocorreu um erro de I/O (Input/Output) no cassete, na impressora ou na operação de CRT; ou seja, há um erro na entrada ou saída de dados. Este erro é fatal e isso significa que o BASIC não pode encontrá-lo.

```
Direct statement in file
```

Termo no Modo Direto foi Encontrado no Arquivo

Um termo direto foi encontrado carregando um arquivo no formato ASCII. O LOAD foi terminado.

```
Division by zero
```

Divisão por Zero

Uma divisão por zero, foi encontrada em uma expressão.

```
FIELD overflow
```

Overflow em FIELD

Tentativa de inserir um FIELD num espaço muito pequeno de bytes.

```
File already open
```

Arquivo Já Aberto

Uma saída seqüencial, no modo OPEN, foi o resultado de arquivo que já havia sido aberto ou, então, um termo KILL abrindo o mesmo tipo de arquivo.

```
File not found
```
Arquivo Não Encontrado

Há um termo LOAD, KILL ou OPEN referente a um arquivo, que não existe na memória.

```
File not open
```
Arquivo Não Aberto

O arquivo especificado em um termo PRINT#, INPUT#, etc. não havia sido aberto.

```
Illegal direct
```
Modo Direto Usado Ilegalmente

Um comando que é usado no modo indireto foi introduzido no modo direto.

```
Illegal function call
```
Função Ilegal

Tentativa de executar uma operação usando um parâmetro ilegal.

```
INPUT past end
```
INPUT Depois do Final

Um termo INPUT foi executado depois de todos os dados de um arquivo (ou arquivo nulo) serem lidos. Evita-se este erro usando a função EOF para encontrar o final do arquivo.

```
Internal error
```
Erro Interno

Mau funcionamento interno.

```
Line buffer overflow
```
Linha do Buffer em Overflow

Uma linha introduzida possui caracteres demais.

```
Missing operand
```
Falta de Operando

Tentativa de operação sem o fornecimento de um dos operandos necessários.

```
NEXT without FOR
```
NEXT sem FOR

Comando NEXT utilizado sem uma instrução FOR correspondente.

```
No RESUME
```
Falta RESUME

Um erro encontrado na rotina foi introduzido porém, não contém o termo RESUME.

```
Out of DATA
```
Insuficiência de Dados

Uma instrução READ foi executada sem encontrar nenhum dado para ler.

```
Out of memory
```
Insuficiência de Memória

Toda memória disponível foi utilizada ou reservada.

```
Out of string space
```
Fora do Espaço para String

O espaço reservado para as variáveis *strings* foi excedido na memória.

Overflow

Sobrecarga

A magnitude de um número é muito grande para o computador.

Redimensioned array

Redimencionamento

Dois termos DIM foram usados para dimensionar a mesma matriz.

RESUME without error

RESUME sem Erro

Um termo RESUME foi encontrado antes de um erro colocado na rotina.

RETURN without GOSUB

RETURN sem GOSUB

Uma instrução RETURN foi encontrada sem que um GOSUB fosse executado antes.

Sequential I/O only

Somente Entrada/Saída Seqüencial

Um termo de acesso randômico está distribuindo o arquivo de forma seqüencial.

String formula too complex

Fórmula de String Muito Complexa

Uma expressão *string* está muito longa ou muito complexa. A expressão deve ser dividiva em expressões menores.

String too long

String Muito Longa

Tentativa de criar uma *string* com mais de 255 caracteres.

Subscript out of range

Sub-índice Fora de Faixa

Tentativa de usar um índice não definido para uma variável indexada.

Syntax error

Erro de Sintaxe

Ocorre quando uma palavra do vocabulário BASIC é escrita de forma incorreta ou quando se faz uso indevido da pontuação.

Type mismatch

Atribuição Ilegal

Para um nome de uma variável *string* foi atribuído um valor, ou vice-versa.

Para uma função, que aguarda um argumento numérico foi dado um argumento *string*, ou vice-versa.

Undefined line number

Número de Linha Não Definido

Uma linha contendo um GOTO, GOSUB, IF...THEN...ELSE, está se referindo a outra linha que é inexistente.

Undefined user function

Uso Indefinido da Função

A função FN foi chamada antes de ser definida pela instrução DEF FN.

Unprintable error (23)

Erro Não Imprimível

Não há mensagem de erro disponível para a condição existente. Isto geralmente acontece quando uma mensagem ERROR aparecer, para um código de erro inválido ou indefinido.

Unprintable error

Erros Não Imprimíveis

Estes códigos não têm definição. Devem ser reservados para futuras expansões do BASIC.

Unprintable error (6Ø-255)

Erro Não Imprimível

Estes códigos não possuem definição. Costuma-se usá-los para definir códigos pessoais de erro.

Verify error

Erro de Verificação

O programa que está sendo executado não está coerente com o programa gravado no cassete.

APÊNDICE D

MAPA DA MEMÓRIA

O EXPERT possui 32 Kbytes de memória ROM (pré-gravada na fábrica, apenas para leitura) e 80 Kbytes de memória RAM (para escrita e leitura). O microprocessador Z80-A pode acessar diretamente 64 Kbytes de memória RAM e é em parte dessa área que o usuário pode trabalhar. Outros 16 Kbytes (VRAM) são acessados pelo microprocessador TMS-9128NL, específico para controlar o vídeo.

Tanto a ROM como a RAM são divididas em páginas para agilizar o acesso aos seus endereços. Cada página de memória tem 16 Kbytes.

A numeração dos endereços da ROM é sempre a mesma, independentemente de se estar ou não usando expansões, interfaces ou cartuchos. Ela se sobrepõe à parte da numeração da RAM e é por isso que apenas 28,8 Kbytes desta podem ser acessados diretamente pelo BASIC.

O MSX possui quatro *slots*, sendo dois internos (0 e 2) e dois externos (1 e 3). Os dois *slots* internos são ocupados pela ROM (*slot* 0) e pela RAM (*slot* 2). Os *slots* externos podem ser conectados a cartuchos, interfaces, etc...

O *slot* 1 tem prioridade sobre o *slot* 3, isto é, se existir um cartucho ligado em cada um deles, prevalecerá a operação do que está no *slot* 1. A conexão do *slot* 1 é a marcada com CARTRIDGE A. O *slot* 3 possui duas conexões: uma dianteira marcada com CARTRIDGE B e outra traseira marcada com BUS EXPANSION. Salvo indicação em contrário, eles não devem ser utilizados simultaneamente.

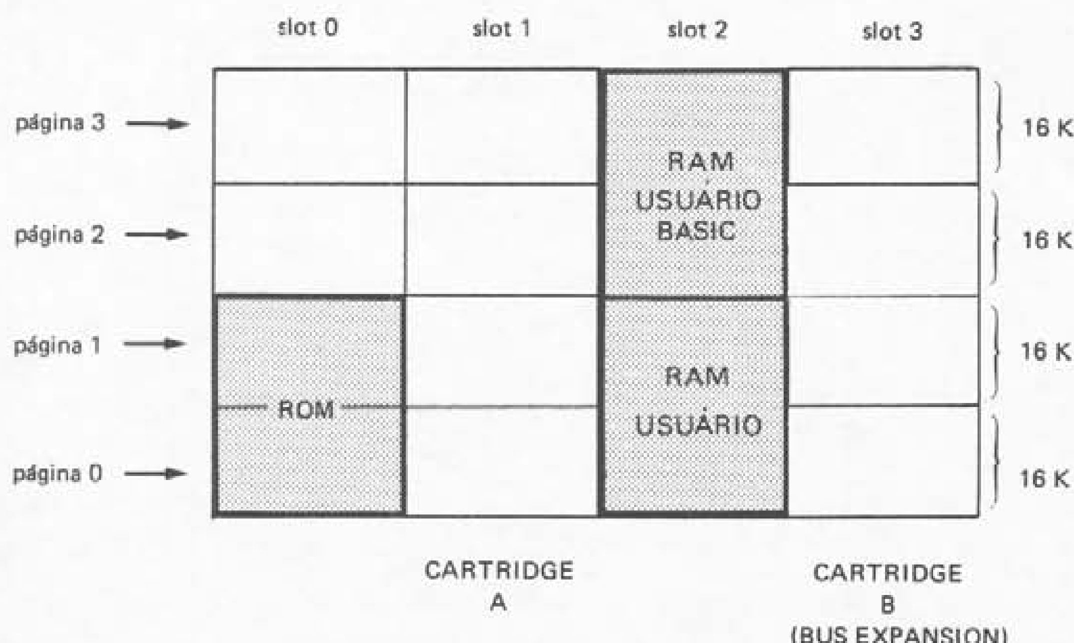


figura D.1 (mapa da memória)

A RAM acessível pelo usuário que programa em BASIC é constituída pelas duas páginas superiores do *slot* 2. As duas páginas inferiores podem ser acessadas mudando-se o status do PPI (veja APROFUNDANDO-SE NO EXPERT, desta mesma editora, para maiores detalhes).

O esquema de distribuição da RAM (usuário BASIC) está indicado na figura D.2.

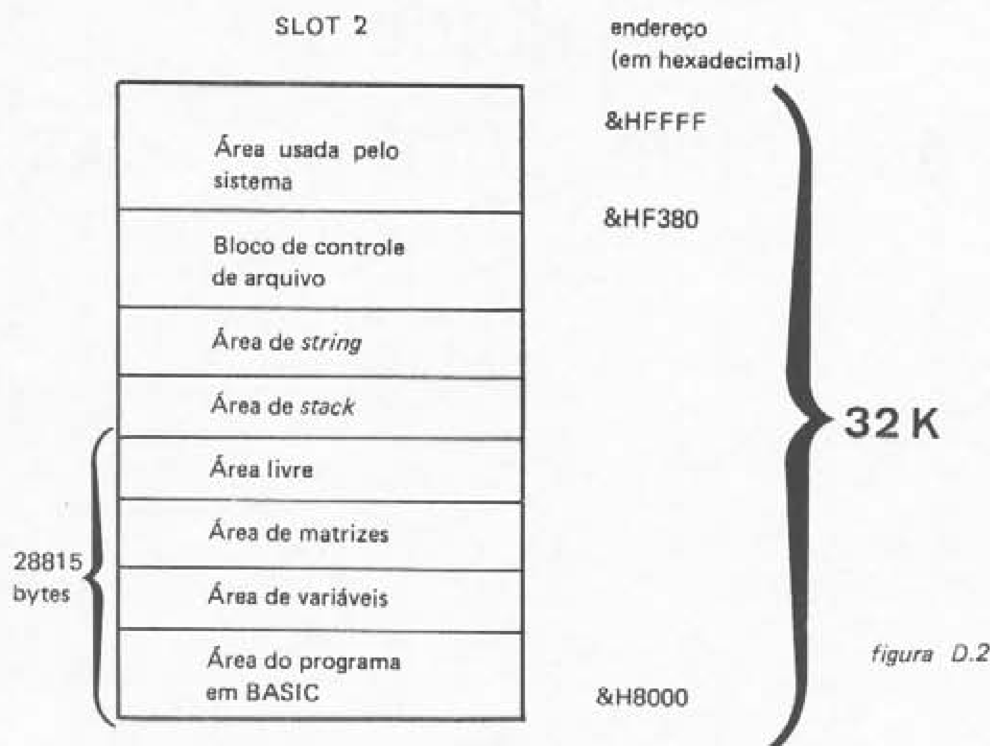


figura D.2

A seguir, uma explicação rápida sobre cada uma das regiões indicadas na fig. D.2:

ÁREA DO PROGRAMA: área onde serão armazenados programas com seus respectivos números de linha.

ÁREA DE VARIÁVEIS: armazena dados numéricos e os *"pointers"* para os dados alfanuméricos.

ÁREA LIVRE: é a área não utilizada.

ÁREA DE STACK: área utilizada para guardar endereços de retorno quando executadas as instruções FOR-NEXT ou GOSUB-RETURN.

ÁREA DE STRING: área utilizada para armazenar strings que englobem variáveis alfabéticas e conjunto de variáveis. O tamanho é definido pela instrução CLEAR. Caso não haja definição por tal instrução, será reservada uma área de 200 bytes.

BLOCO DE CONTROLE DE ARQUIVOS: área utilizada para entrada/saída de arquivos. O tamanho é determinado pela instrução MAXFILES.

APÊNDICE E

ESPECIFICAÇÕES TÉCNICAS

Microprocessador		Z80A (3,58MHz)
Memória residente	ROM do sistema	32 Kbytes (BASIC-MSX)
	RAM	80K Bytes - 64K Bytes usuário - 16K Bytes vídeo
Linguagem de Programação		BASIC-MSX e Linguagem de Máquina Z80
Vídeo	CDP	TMS-9128NL - Processador de Vídeo
	Texto 1	32 colunas x 24 linhas (condição inicial 29 colunas)
	Texto 2	40 colunas x 24 linhas (condição inicial 40 colunas)
	Gráficos	256 x 192 pontos (horizontal x vertical)
	Sprite	32 níveis
	Cor	16 cores
Som	PSG	AY-3-8910-A - Sintetizador de som programável
	Características	08 oitavas, 03 vozes, canal adicional de ruído
Teclado	Disposição das teclas	89 teclas para pontuação, acentuação, caracteres: do Português, gráficos, 10 funções programáveis
	Modalidades	06 modalidades (níveis), 256 endereçamentos de caracteres residentes
Interface residente	Monitor de Som	Alto falante interno com controle de volume externo
	Gravador cassete/ Data Corder	Processo FSK 1200/2400 baud
	Impressora	Padrão tipo Centronics
Conexões externas	Comunicações	Padrão RS232 - Modem padrão Telebrás - Videotexto (opcionais)
	Interface/Cartuchos	02 slots frontais p/ cartuchos padrão MSX (memória virtual, interfaces, aplicativos, games, etc.)
	Barramento de expansão	01 conector, 50 pinos para expandir o sistema, conectar periféricos, etc.
	Vídeo Monocromático	01 conector tipo RCA p/ monitor de vídeo (1,2 VPP/ 75 ohms)
	RGB Analógico	01 conector circular 08 vias (sinal RGB 1 VPP/75 ohms; sinal sincronismo 0,7 VPP/75)
	TV/Monitor a cores	Conector RCA VIDEO COLOR - Saída p/ monitor de vídeo colorido PAL-M Conector RCA VIDEO RF-OUT - Saída p/ antena de TV, através de cabo RF ligado a comutador manual TV/COMPUTADOR. Chave p/ seleção do canal (3 ou 4)

	Saída Som	01 conector tipo RCA - saída misturador (03 canais) interno
	Gravador cassete/data corder	01 conectador circular 05 vias sinais + controle
	Entrada-saída de controles/joystick	02 conectores frontais tipo D9 (09 pinos), 01 p/ cada controlador e/ou joystick
	Impressora	01 conector tipo finger duplo 26 pinos
	Tomadas suplementares	02 tomadas suplementares p/ alimentação de periféricos (controladas pela chave liga/desliga da CPU) carga máxima. 100 VA p/ tomada. Conector e cabo alimentador de 6 V p/ DATA CORDER.
Alimentação	Tensão da rede	AC 120V/240V 60 Hz (chave p/ comutação da tensão da rede)
	Consumo (somente CPU)	Máximo 30 VA
	Consumo (c/ todos periféricos)	Máximo 42 VA
Acessórios	Cabos	01 cabo RF 75 ohms tipo RCA-RCA 01 cabo 05 vias - P1, P2, P2 (REM, AUX, EAR) 01 cabo P4/P4 - Alimentação do DATA CORDER
	Etiquetas	01 conj. de etiquetas de funções p/ titulação pelo usuário das funções programáveis
	Fusíveis complementares	02 fusíveis: 01 p/ 120V, 01 p/ 240V
	Manuais	02 livros: 01 DOMINANDO O EXPERT, 01 LINGUAGEM BASIC MSX. 01 Resumo de Operações do EXPERT.

PRINCIPAIS LSI DO PADRÃO MSX

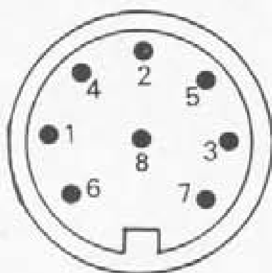
*CPU – Z80 ou equivalente compatível
*VDP – TMS-9128 ou equivalente
*PSG – AY-3-8910 ou equivalente
*PPI – I-8255 ou equivalente
*ROM – MSX BASIC 32 KBytes

APÊNDICE F

PINAGEM

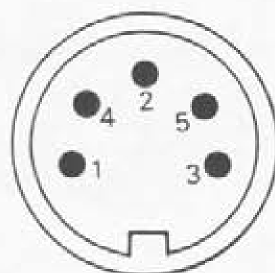
Pinagem RGB

1	Compose sync
2	terra
3	R
4	B
5	G
6	+Vcc
7	Y
8	áudio



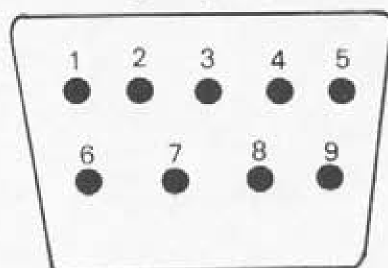
Pinagem do cassete

1	REM
2	Blindagem de dados
3	REM
4	Entrada de dados
5	Saída de dados



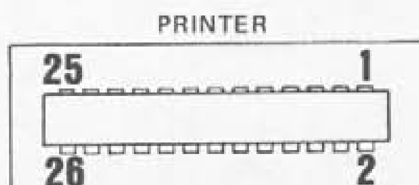
Pinagem Joystick

1	frente
2	trás
3	esquerda
4	direita
5	+5V
6	disparador 1
7	disparador 2
8	saída
9	(GND) terra



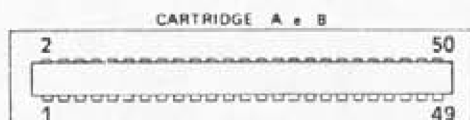
Pinagem da printer

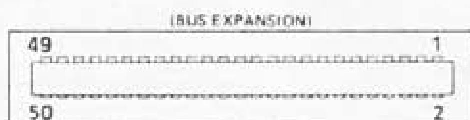
2 a 24	Terra
19, 23, 25, 26	Sem conexão
1	$\overline{\text{STROBE}}$
3	D0
5	D1
7	D2
9	D3
11	D4
13	D5
15	D6
17	D7
21	BUSY



PINAGEM DOS SLOTS E DO BUS EXPANSION

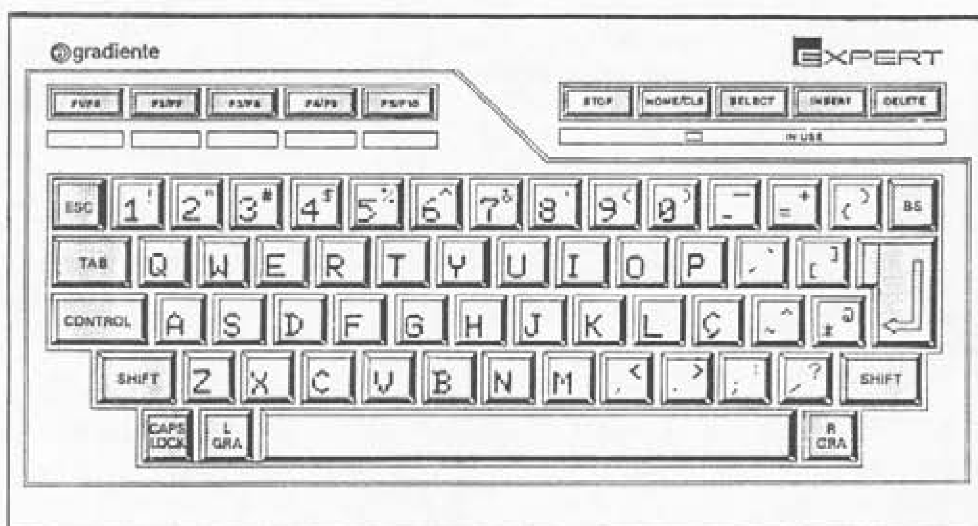
Pino	I/O	NOME	DESCRIÇÃO
01	O	$\overline{CS1}$	*Chip select* da ROM 4000-7FFF
02	O	$\overline{CS2}$	*Chip select* da ROM 8000-BFFF
03	O	$\overline{CS1/2}$	*Chip select* da ROM 4000-BFFF
04	O	$\overline{SLTSL}$	Seleção do *slot*
05	—	reserv.	Reservado para uso futuro
06	O	$\overline{RSFH}$	Sinal de *refresh*
07	I	$\overline{WAIT}$	Sinal de *wait* para CPU
08	I	$\overline{INT}$	Sinal para requisitar interrupção
09	O	$\overline{M1}$	Sinal indicador do ciclo de *fetch*
10	I	$\overline{BUSDIR}$	Controle da direção do *buffer* do *bus* de dados
11	O	$\overline{IORQ}$	Sinal de requisição de I/O
12	O	$\overline{MERQ}$	Sinal de requisição de memória
13	O	$\overline{WR}$	Sinal de escrita
14	O	$\overline{RD}$	Sinal de leitura
15	O	$\overline{RESET}$	Sinal de *reset* do sistema
16	—	reserv.	Reservado para uso futuro
17	O	A 9	*Bus* de endereços
18	O	A15	"
19	O	A11	"
20	O	A10	"
21	O	A 7	"
22	O	A 6	"
23	O	A12	"
24	O	A 8	"
25	O	A14	"
26	O	A13	"
27	O	A 1	"
28	O	A 0	"
29	O	A 3	"
30	O	A 2	"
31	O	A 5	"
32	O	A 4	"
33	I/O	D 1	*Bus* de dados
34	I/O	D 0	"
35	I/O	D 3	"
36	I/O	D 2	"
37	I/O	D 5	"
38	I/O	D 4	"
39	I/O	D 7	"
40	I/O	D 6	"
41	—	GND	Terra
42	—	CLOCK	*Clock* da CPU de 3 575 611 Hz
43	—	GND	Terra
44	—	SW 1	Interruptor de proteção da retirada e inserção do cartucho, ou Bus Expansion
45	—	+ 5V	+ 5V
46	—	SW 2	Interruptor de proteção na retirada e inserção do cartucho, ou Bus Expansion
47	—	+ 5V	+ 5V
48	—	+ 12V	+ 12V
49	I	SOUNDIN	*Input* de Som
50	—	-12V	-12V

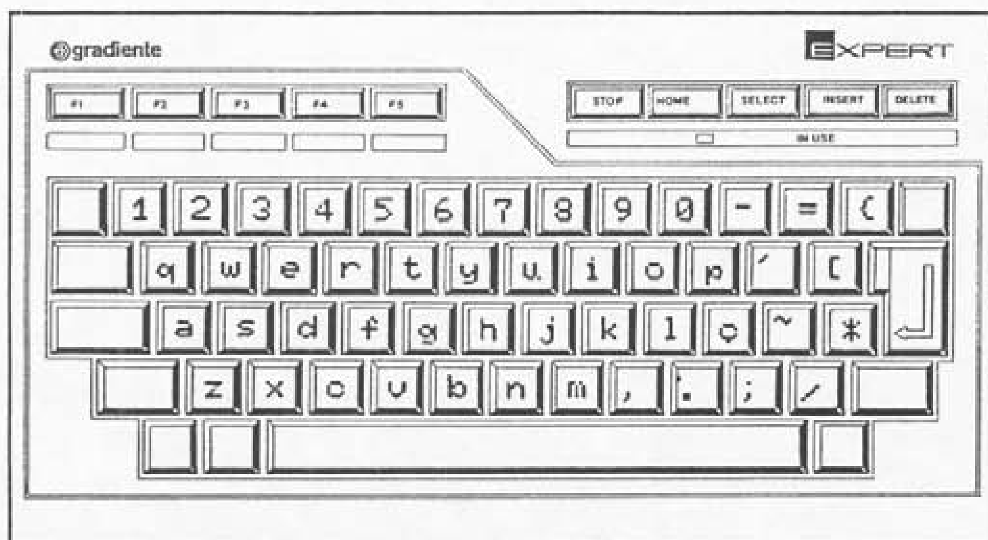


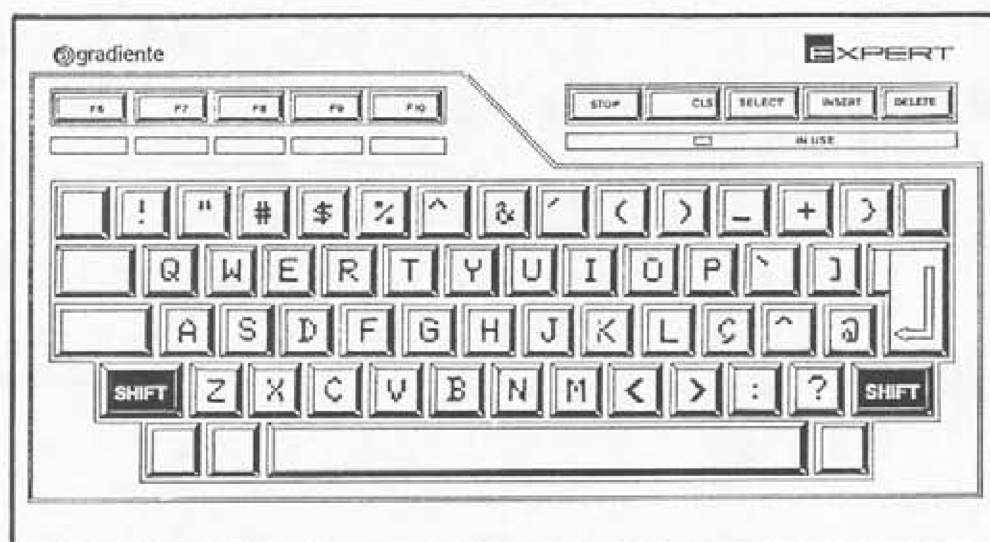


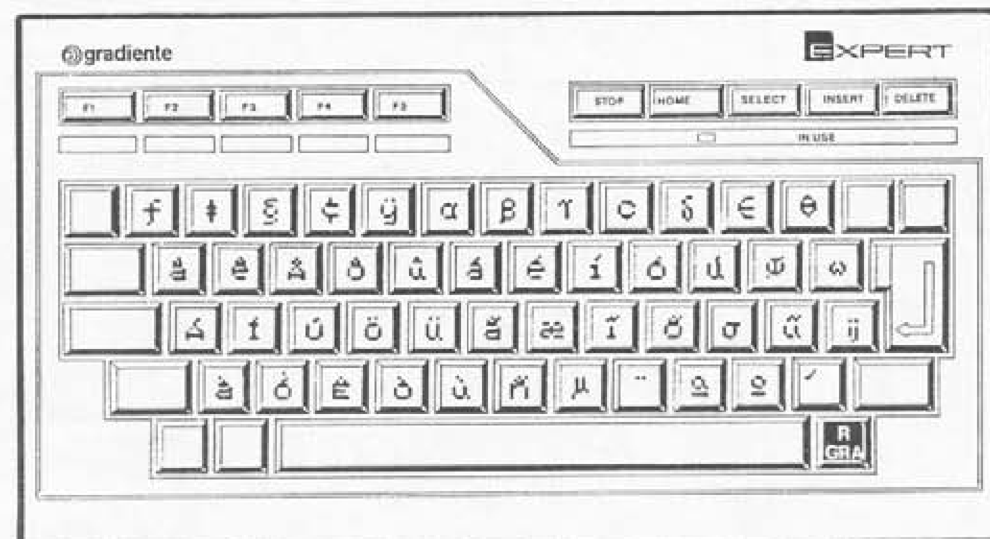
APÊNDICE G

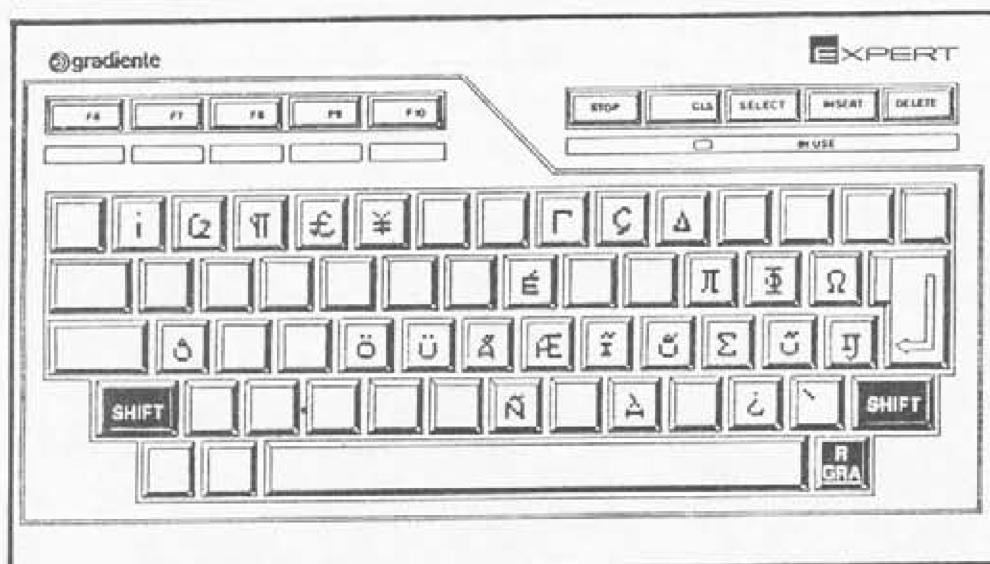
TECLADOS

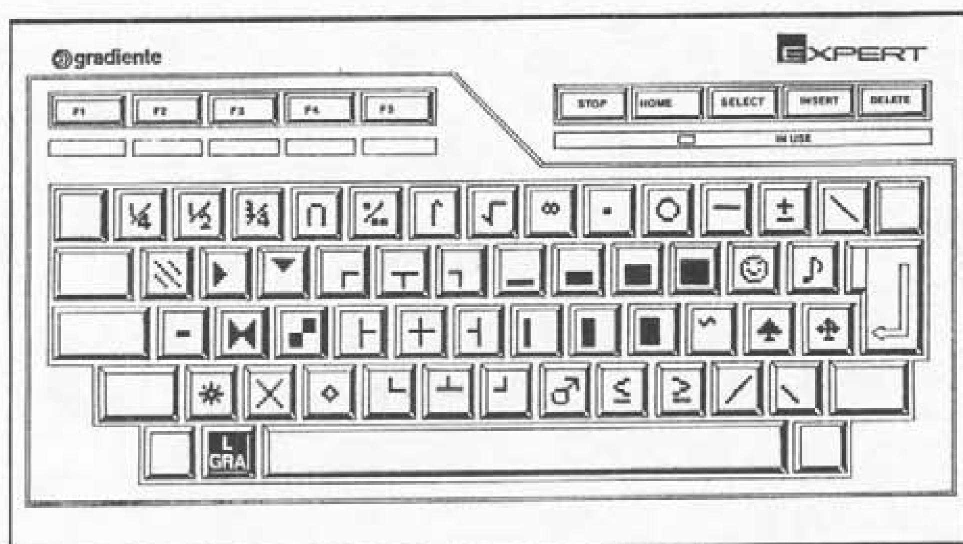


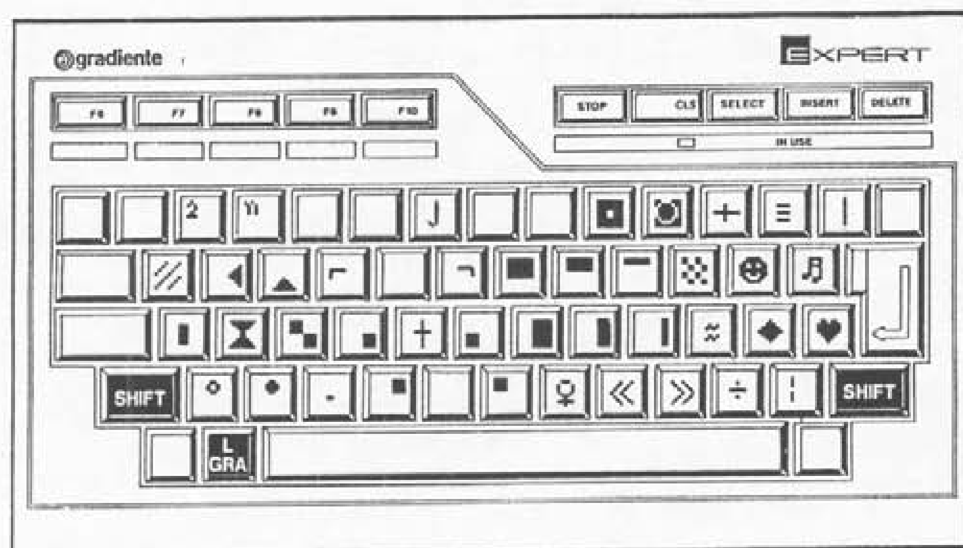



gradiente
EXPERT
STOP
CLS
SELECT
INSERT
DELETE
IN USE
SHIFT
SHIFT


gradiente
EXPERT
STOP
HOME
SELECT
INSERT
DELETE
IN USE
R
GRA


gradiente
EXPERT
STOP
CLS
SELECT
INSERT
DELETE
IN USE
SHIFT
SHIFT
R
GRA


gradiente
EXPERT
F1
F2
F3
F4
F5
STOP
HOME
SELECT
INSERT
DELETE
IN USE
L
GRA


gradiente
EXPERT
F6
F7
F8
F9
F10
STOP
CLS
SELECT
INSERT
DELETE
IN USE
SHIFT
SHIFT
L
GRA

APÊNDICE H
GLOSSÁRIO

Neste glossário você irá encontrar algumas palavras de uso corrente entre os programadores no Brasil. Muitas delas são de origem inglesa, outras são termos técnicos ou de uso restrito a uma certa área de atividade e nem sempre podem ser encontradas em dicionários.

A maioria dessas palavras foram usadas com intuito de familiarizar o leitor com termos que ele irá encontrar freqüentemente em textos ligados à área da informática. Apesar de suscintas, as explicações são suficientes para uma primeira leitura.

ÁLGEBRA BOOLEANA — Álgebra usada para matematizar operações lógicas. Foi formalizada pela primeira vez pelo matemático inglês George Boole.

ALFANUMÉRICO — Termo genérico que designa todos os caracteres que representam letras ou números.

ALEATÓRIO — Ao acaso. Todo fenômeno que não tem suas causas determináveis. Ver RANDÔMICO.

ARRAY — É uma variável indexada (matriz ou tabela).

ASCII — *American Standard Code for Interchange Information.* É um código padrão utilizado na maioria dos microcomputadores.

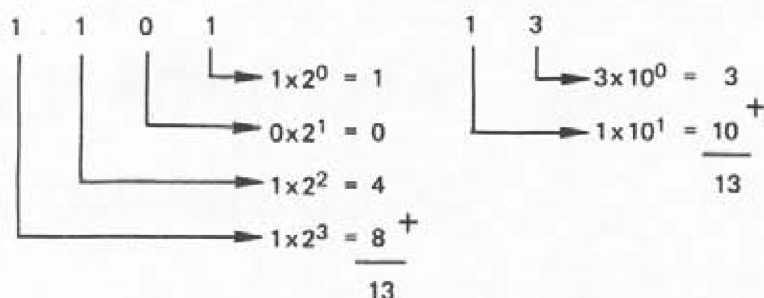
ASSEMBLER — Programa que permite a programação em linguagem *Assembly*.

ASSEMBLY — É a linguagem mais fundamental depois da própria linguagem da máquina. É extremamente complexa a programação direta com códigos de máquina e, para evitá-la, associou-se um código mnemônico à cada instrução de máquina. O *Assembly* é o conjunto desses códigos mnemônicos.

BASIC — É uma das mais difundidas linguagens de programação (além de ser uma das mais simples!). Existem diversas versões do BASIC sendo que uma das mais completas é a usada pelo padrão MSX.

BAUD — Unidade de velocidade de transmissão de informação. A menor unidade de informação possível é o bit. Um baud é um bit por segundo. Por exemplo, se um sistema transmite dados com a velocidade de 100 bauds, significa que ele transmite 100 bits em cada segundo.

BINÁRIO — Sistema de numeração que usa apenas os algarismos 0 e 1. Por exemplo, o número 13 escrito no sistema binário será 1101.



BIT — É a unidade mais fundamental de informação.

BUFFER — Área de memória onde são, temporariamente, armazenados dados.

BUG — Termo que designa um problema num programa. Poderia ser traduzido como "grilo", ou ainda "galho".

BUS — O mesmo que transmissor. Emprega-se esse termo geralmente para designar um conjunto de terminais que servem para transmissão ou recepção de informações. Barra de transmissão ou recepção de informação.

BYTE — É uma unidade lógica de informação. Inicialmente designava um conjunto arbitrário de bits porém, devido à padronização imposta pelo uso, designa um conjunto de 8 (oito) bits.

CARREGAR — Transferir informações de uma memória (geralmente externa: gravador, *floppy-disk*, etc...) para a memória RAM.

CHIP — É um componente eletrônico no interior do qual existem circuitos complexos destinados à execução de operações específicas.

CLOCK — É o gerador da freqüência de operação do microprocessador, ou seja, do número de instruções elementares que ele pode realizar em cada segundo.

COMANDO — Ordem a ser executada imediatamente.

COMPILADOR — Programa que traduz conjuntos completos de instruções, de uma linguagem para outra.

CPU — Unidade Central de Processamento. É a parte mais fundamental de um computador. Nos micros geralmente ela é constituída por um único *chip*. É a CPU quem gerencia todo o funcionamento do computador, distribuindo tarefas às partes secundárias.

CRT — *Catode Ray Tube* — Tubo de raios catódicos ou TV. É a designação da tela em modo texto como dispositivo.

CURSOR — Indicador da posição em que será escrito o próximo caractere digitado.

DATA	Dados (em latim). É a instrução BASIC que permite armazenar dados em linhas de programas.
DATA BUS	Barra de endereçamento de dados.
DEBUG	Eliminação de erros de um programa.
DELETAR	Eliminar caracteres à direita do cursor ou linhas de um programa.
DISK	Disco onde são gravadas informações através de processos eletromagnéticos.
DOS	*Disk Operation Sistem.* Sistema de operação de discos.
DRIVER	É um termo usado com vários significados. Pode ser imaginado como o gerente de uma certa atividade. Por exemplo, DISK DRIVER é o controlador (gerente) de discos flexíveis.
DUMP	Termo que designa a verificação e apresentação através de um meio de saída do conteúdo de todas as variáveis ou arquivos de um programa.
EDITOR	Rotina destinada a facilitar a escrita numa linguagem e eventuais correções da mesma.
FILE	Arquivo. Informações armazenadas em memória externa auxiliar (geralmente, *floppy-disk*).
HARDWARE	É a parte eletrônica de um sistema. Pode ser entendido como arquitetura de construção da máquina.
HEXADECIMAL	Sistema de numeração que usa dezesseis dígitos para representar os números. Os dígitos são: 0,,1,2,3,4,5,6,7,8,9,A,B,C,D,E e F. O número decimal 59 equivale a 3B em hexadecimal.

```
3  B                              5  9
|  └──►11x16⁰ = 11                |  └──►9x10⁰ =  9
|                 +               |                 +
└─────► 3x16¹ = 48                └─────►5x10¹ = 50
               ────                              ────
                59                                59
```

INPUT	É a entrada de um sistema.
INTERFACE	Circuito que permite a interligação e comunicação entre dois elementos de um sistema.
INTERPRETADOR	Programa que traduz, instrução por instrução, de uma linguagem para outra.
INSTRUÇÃO	É uma ordem para ser executada num determinado momento dentro de uma dada seqüência.
I/O	Input/Output. Entrada e saída de um sistema.

JOYSTICK — Alavanca de controle manual geralmente utilizada em jogos de ação.

KBYTE — 1024 bytes.

LOCAÇÃO — Posição provisória de um programa ou de dados na memória do computador.

LOAD — Comando que carrega informações de uma memória externa auxiliar para a memória RAM.

LOOP — Laço. Parte de um programa que é executada várias vezes seguidas.

LSI — *Large Scale Integration.* Integração em larga escala. São circuitos integrados (*chip's*) equivalentes a milhares de componentes discretos.

LINGUAGEM DE MÁQUINA — É a linguagem que o microprocessador entende. Suas instruções são seqüências de pulsos elétricos e podem ser imaginadas como seqüências de 0's (ausência de sinal) e 1's (presença de sinal). A informação é codificada, portanto, no sistema binário.

MATRIZ — O significado mais usual de matriz é o matemático. Uma matriz é uma tabela em que cada elemento é distinguido dos demais por índices (ou coordenadas). O número de índices necessário para identificar um elemento de uma matriz é a sua dimensão. Por exemplo, a dimensão da matriz a seguir é dois, pois precisamos de 2 números para identificar cada um de seus elementos. O único número 8 dessa matriz está na posição (2,4), linha 2 e coluna 4. Veja também ARRAY.

colunas

linhas	1	3	5	5	7
	2	3	5	8	6
	3	4	1	1	9

MEMÓRIA — É qualquer elemento armazenador de informações.

MENU — Apresentação de opções a serem escolhidas pelo usuário de um programa.

MICROPROCESSADOR — É a parte mais fundamental de um computador. É ele quem gerencia o fluxo de informação através da máquina. Veja CPU.

MNEMÔNICOS — São códigos da linguagem ASSEMBLY.

MODEM — *MODulator DEModulator.* Dispositivo que transforma dados em sinais que podem ser transmitidos ou recebidos por linha telefônica. Equipamento que interliga um microcomputador a um sistema telefônico.

OCTAL — Sistema de numeração baseado no número 8. O número decimal 33 equivale ao número octal 41.

$$4\ 1: \quad 1 \times 8^0 = 1,\quad 4 \times 8^1 = 32,\quad 1 + 32 = 33$$

$$3\ 3: \quad 3 \times 10^0 = 3,\quad 3 \times 10^1 = 30,\quad 3 + 30 = 33$$

OFF-SET	Deslocamento que deve ser acrescentado a um endereço base para obter outro endereço.
ON-LINE	Modo de operação de um sistema em que a comunicação da máquina com o usuário é direta, imediata e, geralmente, através de vídeo.
OVERFLOW	Sobrecarga.
OUTPUT	É a saída de um sistema.
PADDLE	Controle manual geralmente empregado para jogos. Difere do *joystick* por ser analógico.
PERIFÉRICO	É qualquer elemento que pode ser acrescentado a um sistema, tornando-se parte dele.
PIXEL	Elemento fundamental de impressão no vídeo.
PLOTTER	Periférico destinado a confecção de gráficos e outros tipos de desenhos em papel.
POINTER	Indicador de posição. Ponteiro.
PRINTER	Impressora. Dispositivo que imprime dados enviados pelo computador em papel.
RAM	*Random Access Memory*. Memória de acesso aleatório. É a memória alterável onde são armazenados os programas e dados.
RANDÔMICO	O mesmo que aleatório. Ao acaso.
ROM	É a memória permanente (que não se perde quando o micro é desligado) onde está armazenada a linguagem BASIC e outras informações indispensáveis à operação.
REGISTRO	Elemento de memória interno de um microprocessador. Um registro só é acessado pelo próprio *chip* a que ele pertence.
RETURN	Retorno. É a tecla que envia as informações introduzidas através do teclado para a memória RAM do micro ou que ordena a execução de um comando. É também uma palavra do vocabulário BASIC usada no final de todas as subrotinas de um programa e que devolve o processamento ao programa principal.
RF	Rádio Freqüência. Sinal de freqüência modulada que chega à antena receptora de um aparelho de TV.

RGB — Três sinais já decodificados a partir do sinal de vídeo-composto. Um é o sinal para cor vermelha (*Red*), outro para cor verde (*Green*) e outro para a cor azul (*Blue*).

ROTINA — É uma parte lógica de um programa que pode ser distinguida individualmente.

SCREEN — Tela.

SOFTWARE — Designação genérica de programa. Tudo que é relacionado com a programação faz parte do *software*. Um programa é um *software*.

SPRITE — Máscara que pode ser sobreposta à tela. Figura definida através de *software* que pode ser movimentada e sobreposta a outras figuras na tela.

SUB-ROTINA — É uma parte de um programa que pode ser executada várias vezes em situações distintas.

SINTAXE — Forma correta de escrever.

SISTEMA OPERACIONAL — Sistema lógico de gerenciamento de uma máquina. É o responsável pela operação automática da máquina, independentemente do operador.

SLOT — É um encaixe para conexão de periféricos diversos, *drivers*, cartuchos, expansões, etc.

STRING — Nome que se dá a qualquer seqüência de caracteres introduzida no micro. Quando são usadas após comandos (como PRINT, LET, INPUT, etc...) devem estar entre aspas.

TERMINAL — Meio físico ou lógico através do qual saem ou entram informações num sistema.

TABELA VERDADE — Tabela usada na análise de operadores lógicos e que os determinam completamente. Por exemplo, a tabela do operador XOR é a seguinte:

E2 \ E1	0	1
0	0	1
1	1	0

TOUCH PAD — Painel sensível a toque.

UTILITÁRIOS — São programas que agem sobre outros programas facilitando ao usuário a edição dos mesmos.

VARIÁVEL — É um elemento armazenador de dados.

VÍDEO-COMPOSTO — Sinal da moduladora da RF. A partir deste sinal são obtidos os sinais RGB.

APÊNDICE I

FUNÇÕES TRIGONOMÉTRICAS E HIPERBÓLICAS

O BASIC MSX permite calcular diretamente as funções SIN (seno), COS (cosseno), TAN (tangente) e ATN (arco-tangente) onde os ângulos são medidos em radianos.

Para as outras funções trigonométricas e hiperbólicas, o cálculo deve ser precedido da definição de função. Por exemplo, sabendo-se que a função secante é o inverso do cosseno, podemos calculá-la para um ângulo X qualquer (em radianos) usando o seguinte programa exemplo:

```
10 DEF FN SEC(X) = 1/COS(X)
20 INPUT X
30 PRINT FN SEC(X)
```

A correlação entre as funções a serem definidas e as residentes no BASIC MSX é dada na tabela a seguir.

```
SEC(X)       = 1/COS(X)
CSC(X)       = 1/SIN(X)
COT(X)       = 1/TAN(X)
ARCSIN(X)    = ATN(X/SQR(-X*X+1))
ARCCOS(X)    = -ATN(X/SQR(-X*X+1))+1.5708
ARCSEC(X)    = ATN(SQR(X*X-1))+(SGN(X)-1)*1.5708
ARCCSC(X)    = ATN(1/SQR(X*X-1))+(SGN(X)-1)*1.5708
ARCCOT(X)    = -ATN(X)+1.5708
SINH(X)      = (EXP(X)-EXP(-X))/2
COSH(X)      = (EXP(X)+EXP(-X))/2
TANH(X)      = -EXP(-X)/(EXP(X)+EXP(-X))*2 + 1
SECH(X)      = 2/(EXP(X)+EXP(-X))
CSCH(X)      = 2/(EXP(X)-EXP(-X))
COTH(X)      = EXP(-X)/(EXP(X)-EXP(-X))*2 + 1
ARCSINH(X)   = LOG(X+SQR(X*X+1))
ARCCOSH(X)   = LOG(X+SQR(X*X-1))
ARCTANH(X)   = LOG((1+X)/(1-X))/2
ARCSECH(X)   = LOG((SQR(-X*X+1)+1)/X)
ARCCSCH(X)   = LOG((SGN(X)*SQR(X*X+1)+1)/X)
ARCCOTH(X)   = LOG((X+1)/(X-1))/2
```

Se você quiser obter mais informações sobre LITERATURA específica para o MSX escreva para a ALEPH Publicações e Assessoria Pedagógica Ltda. C.P.: 20.707 - São Paulo -SP





Este livro foi impresso com
fotolitos fornecidos pela Editora
Gráfica Palas Athena
Associação "Palas Athena" do Brasi'
Rua José Bento, 384
Fone: 279-6288 — CEP 01523
Cambuci — São Paulo

Editora Aleph

gradiente

CAIXA POSTAL 30.318 - SÃO PAULO - SP
PRODUZIDO NA ZONA FRANCA DE MANAUS
PRINTED IN BRAZIL
901-01940-000 06-2648-001

AS CORES DO EXPERT

0 incolor
1 preto
2 verde
3 verde claro
4 azul escuro
5 azul claro
6 vermelho escuro
7 ciano
8 vermelho
9 vermelho claro
10 ouro
11 amarelo
12 verde musgo
13 magenta
14 cinza
15 branco